UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona (FIB)

Development of a protein quantification algorithm for data analysis in the field of proteomics

Ponent:
Lluís Antonio Belanche Muñoz

Autora:
Ariadna Llovet Soto

Departament:
LSI

Titulació:
Enginyeria superior en informàtica

Directora:
Judit Villén

Institució:
University of Washington

Barcelona, January 14, 2014

Contents

1 Introduction **3**
- 1.1 Project definition . . . 3
- 1.2 Document structure . . . 4

2 Proteomics Quantification **5**
- 2.1 Mass spectrometry . . . 5
- 2.2 Quantification . . . 7
  - 2.2.1 Concepts . . . 8
  - 2.2.2 Phases . . . 11

3 The proteomics quantification algorithm **12**
- 3.1 Algorithm . . . 12
  - 3.1.1 Labels . . . 13
  - 3.1.2 Peak in time finding . . . 16
  - 3.1.3 Isotopic distribution abundance . . . 22
  - 3.1.4 Noise . . . 32
  - 3.1.5 Peptide treatment . . . 35
  - 3.1.6 Area . . . 70
  - 3.1.7 Output . . . 75
- 3.2 Quantification methods . . . 77
  - 3.2.1 Batch quantification . . . 77
  - 3.2.2 Cross quantification . . . 77
  - 3.2.3 Single quantification . . . 82
- 3.3 Algorithm costs . . . 82

4 Algorithm design **85**
- 4.1 Use cases . . . 85
- 4.2 Software architecture . . . 86
- 4.3 Class diagram . . . 88
- 4.4 Implementation decisions . . . 93

5 The proteomics quantification module **96**
- 5.1 Requirements . . . 96
  - 5.1.1 Functional requirements . . . 96
  - 5.1.2 Non-functional requirements . . . 98
- 5.2 Design . . . 100

5.2.1 Quantification 100
5.2.2 Results view 101
5.3 Implementation 102
5.3.1 Quantification 106
5.3.2 Results view 106

6 Results analysis **109**
6.1 Algorithm 109
6.2 PPM study 112
6.3 Peptides finding 113

7 Future improvements **117**

8 Alternatives **118**

9 Economical analysis **120**

10 Conclusion **122**

11 Glossary **124**

12 Bibliography **128**

13 Annex **131**
13.1 Algorithm user manual 131
13.2 Module user manual 146
13.3 Module ajax actions 168
13.4 Configuration file 187
13.5 Configuration file example 199

1 Introduction

1.1 Project definition

The proteomics quantification target is to obtain information relative to the peptides that can be found in a sample. These samples are composed by peptides, which are extracted from the cell proteins or tissues that will be analyzed. The quantification results are used in order to find the existing differences between these samples. An example could be a comparison between healthy and diseased patients.

This project consists on a proteomic quantification algorithm development for the Villén lab at the University of Washington (UW). The Villén lab wanted to add new features into their current system. Because of the complexity that was representing, they decided to create their own platform for future maintenance and scalability. The quantification algorithm will be a part of this new platform, becoming a new module for the system.

Therefore, this project has two different defined targets:

- The Development of a protein quantification algorithm for data analysis in the field of proteomics.
- To develop a web module in order to interact with the algorithm.

Target	Sub-target
Algorithm	Obtain peak in time
	Isotopic distribution abundance
	Calculate noise
	Peptide treatment
	Area calculation
Module	Execute
	View results
	Download results

Table 1: Project targets.

1.2 Document structure

The main goal of this project is to provide the Villén lab with a new proteomics quantification algorithm. The project has two different parts: the algorithm implementation, and a web-based module development in order to use this algorithm.

At first instance, the proteomics quantification concept will be explained in section 2, showing some information regarding to the mass spectrometry and the proteomics quantification concepts and phases. Following this explanation, the quantification algorithm will be shown in detail in section 3, starting with the explanation of the algorithm itself, continuing with the peptide treatments, and finally showing the algorithm costs. Following the algorithm explanation, the design decisions are explained in section 4, showing the use cases, class diagram... After this, the proteomics quantification module will be exposed in section 5, showing the design and implementation decisions made. The document will end discussing some quantification results (6), enumerating future improvements (7), showing possible alternatives (8), doing an economical analysis (9), and explaining the project conclusions (10).

2 Proteomics Quantification

The aim of proteomic quantification is to obtain quantitative information of all the proteins in a sample.
This quantitative information is useful in order to obtain differences between sample pools, where at least one pool was exposed to a stimulus of interest, allowing, for example, compare samples from healthy and diseased patients. These pools are easily distinguished by mass within a mass spectrometry scan.

2.1 Mass spectrometry

Quantitative proteomics uses the mass spectrometry (MS) technology to detect the protein changes. The mass spectrometry, is the science of displaying the spectra of molecules masses comprising a sample of material.

A mass spectrum is an *intensity* vs. mz plot representing a chemical analysis or distribution of ions by mass. The mz m refers to the molecular or atomic mass and the z refers to the number of elementary charges carried by the ion, i.e. mass-to-charge ratio.

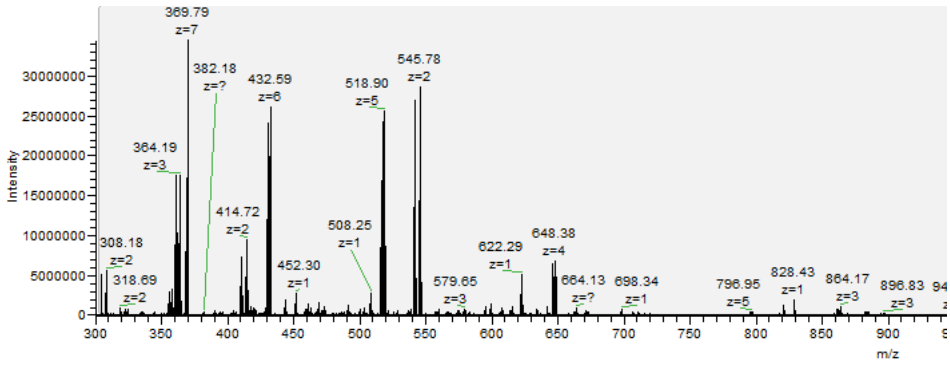


Figure 1: Mass spectra (*intensity* vs mz).

The mass spectrometer works by ionizing chemical compounds to generate charged molecules or molecule fragments and measuring their mass-to-charge ratios.
The mass spectrometer consists on three components: the ionizer, the extractor and the detector. The ionizer converts some portion of a sample

into ions. The extraction system removes ions from the sample and gives them a trajectory which allows the mass analyser to sort the ions by mass-to-charge, this way separates these ions according to their mass-to-charge ratios. The detector, who has a mechanism capable of detecting charged particles, detects the ions, and finally, the produced signal is processed into the previously explained spectra of the relative abundance of ions as function of the mass-to-charge ratio.

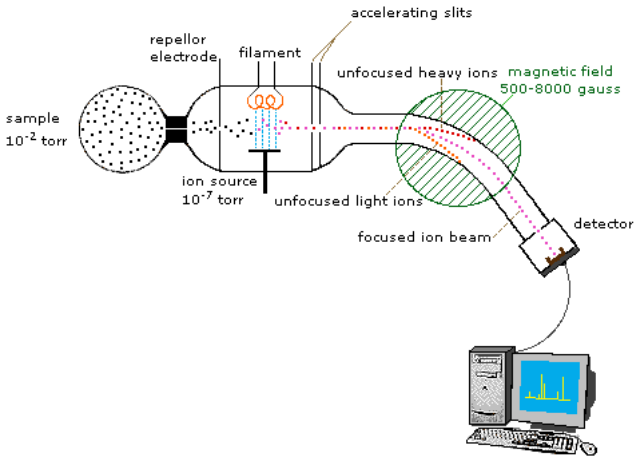


Figure 2: Mass spectrometer flow.

The MS data

The mass spectrometer outputs its data as a set of scans.

The proteomics quantification is interested in two different kind of scans: the level one scans (MS1), and the level two scans (MS2). The MS1 scans contain information of the detected ions in a concrete retention time, while the MS2 scans contain information of a concrete MS1 ion (ion precursor).

This scan detection is done as follows: In the mass spectrometer, an ion pre separation is done using the Liquid Chromatography (LC). The LC has a carbon chain in a column that retains the ions. When in a concrete time (retention time) some of these ions "fall", a MS1 scan is produced with information regarding to these ions (*intensity* vs mz peaks). For a better study, the mass spectrometer stimulates the most intens "fallen" ions, basi-

cally applying gas, in order to fragment them. The resulting daughter ions are analyzed in the MS2 scans, which are precursor scans.

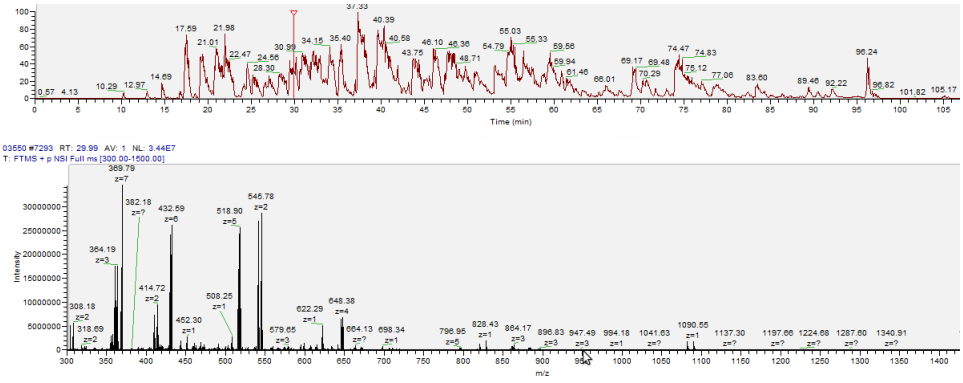


Figure 3: The first image shows MS2 scans along the time (mz vs time), while the second image shows an MS1 scan of a concrete precursor MS2 scan (retention time: 29.99).

2.2 Quantification

The pool introduced in the mass spectrometer, can contain different samples, (healthy, diseased...) labeled usually as Light and Heavy.
The quantification goal is to obtain the mass difference between all the labeled peptides and their labeled counterparts. This way, a disease, like cancer, can be detected.

In this project, the peptide quantification will be done obtaining the area (retention time and intensity shape) ratio between different labels (Light label with its counterparts).

In some cases, a concrete peptide can not be quantified:

- If a labeled peptide is found but its counterpart is not, means that this peptide is not present in the sample, and therefore, its quantification is not interesting for the experiment.
- If the peptide provided is the end of a protein, the quantification of that peptide won't be necessary as no labeled counterparts will be found.

2.2.1 Concepts

In this section, some concepts related to the proteomics quantification will be explained; these concepts are strongly related between them, so in order to provide a better explanation, a basic class diagram will be build step by step.

Peptide
Peptides are short chains of amino acids. Peptides are distinguished from proteins on the basis of size, and as a benchmark can be understood to contain approximately 50 amino acids or less. Proteins consist of one or more polypeptides arranged in a biologically functional way. Peptides have their own mass, charge...

In proteomics quantification, the quantification is done comparing the peptide's area with this same peptide but labeled. The labeled peptide has the same amino sequence acid than the original one, but its mass is different due to amino acid mass changes.

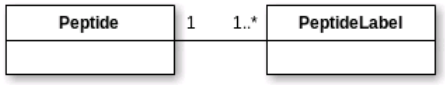


Figure 4: A peptide can have multiple labels, but at least has one: the label usually named as Light, does not have any amino acid mass changes.

Spectra
The peptide's area can be obtained from the Mass Spectrometer's output data. From the previously executed search algorithm, the peptide precursor scan is obtained, i.e the scan where the peptide can be found. From this scan, moving along the retention time (back and forward) in every scan that the peptide is found, its spectra is found too. The spectra is the isotopic distribution of the peptide, understanding the isotopic distribution as a distribution of isotopes where every isotope has $(100-i)$ Carbon 12 atoms and i Carbon 13 atoms, where i is the isotope position in the distribution starting by 0.

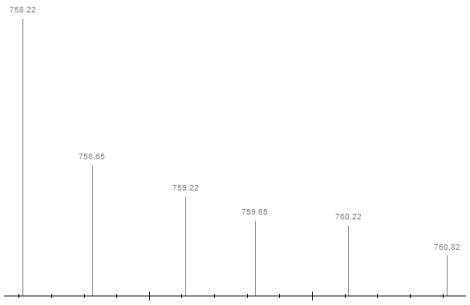


Figure 5: Peptide spectra found in a concrete MS scan.

Every labeled peptide will have their own spectra set. From where, sorted by retention time, the labeled peptide shape can be obtained.

Peak
A spectra is composed by a set of peaks.

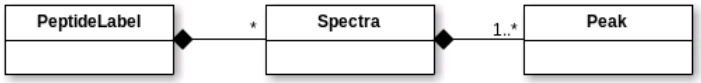


Figure 6: The labeled peptides are composed by a set of spectrums, which at the same time are peak sets.

There are two different kind of peaks: Profile and Centroid.
In profile mode, a peak is represented by a collection of signals over several scans. The advantage of profile data is it is easier to classify a signal as a true peak from noise off the instrument.
In centroid mode, the signals are displayed as discrete mz with zero line widths. The advantage of centroid data is the file size is significantly smaller as there is less information describing a signal.

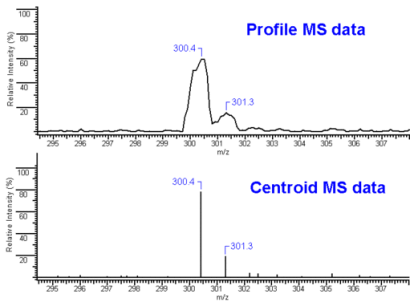


Figure 7: Representation of Profile and Centroid peaks.

In this project only the centroid peaks will be used. A centroid peak is an *intensity* vs mz point that can be found in the MS1 Mass Spectrometer scans.

The peak that has the same mz than the peptide is the monoisotopic spectral peak, the other spectral peaks can be obtained calculating the mass difference between isotopes.

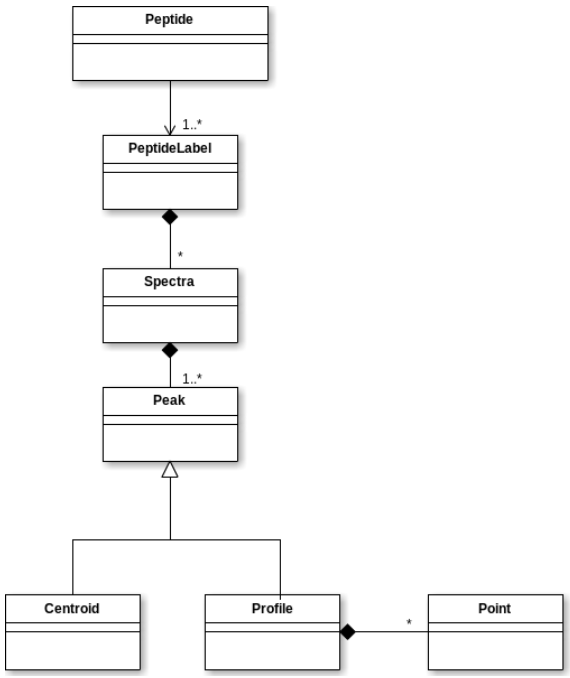


Figure 8: Basic class diagram that shows the proteomic quantification algorithm essential concepts.

2.2.2 Phases

The peptide quantification algorithm can be divided in two parts: the peptide finding and the quantification itself.
The peptide finding contains all the labeled peptides search and checking, while the quantification does the needed peptide treatments and area calculations.

Phases:

- Peptide finding in the mass spectrometer's output file:
    1. Calculate the different labels mz.
    2. Find the different mz's along the time in order to obtain their chromatogram ($intensity$ vs mz along the time).
    3. Calculate the isotope distribution abundance in order to recalculate overlapped intensities.
    4. Calculate the peptide noise.
- Area calculation for every label:
    1. Calculate the peptide quality score.
    2. Find the peptide limits in the chromatogram.
    3. Reconstruct the chromatogram if necessary.
    4. Calculate the peptide area.
- Area ratios calculation.

These phases will be detailed at the next section.

3 The proteomics quantification algorithm

This section is divided in three different subsections. The first and the second part will introduce the algorithm itself and its quantification possibilities respectively, while in the third part the algorithm cost is explained.

3.1 Algorithm

In the proteomics quantification algorithm, a quantification is done for every given peptide. This peptide quantification process is the same for every peptide and can be clearly divided in two parts: the peptide finding and the quantification process. The peptide finding is the part of the algorithm where the peptide (with all the label variants) is searched in the mass spectrometer output file, the found peaks along the time (peak in time) are stored, and its noise and possible intensity overlappings are calculated. Whereas the quantification process does some peak in time treatments, area calculations, and finally the quantification itself.

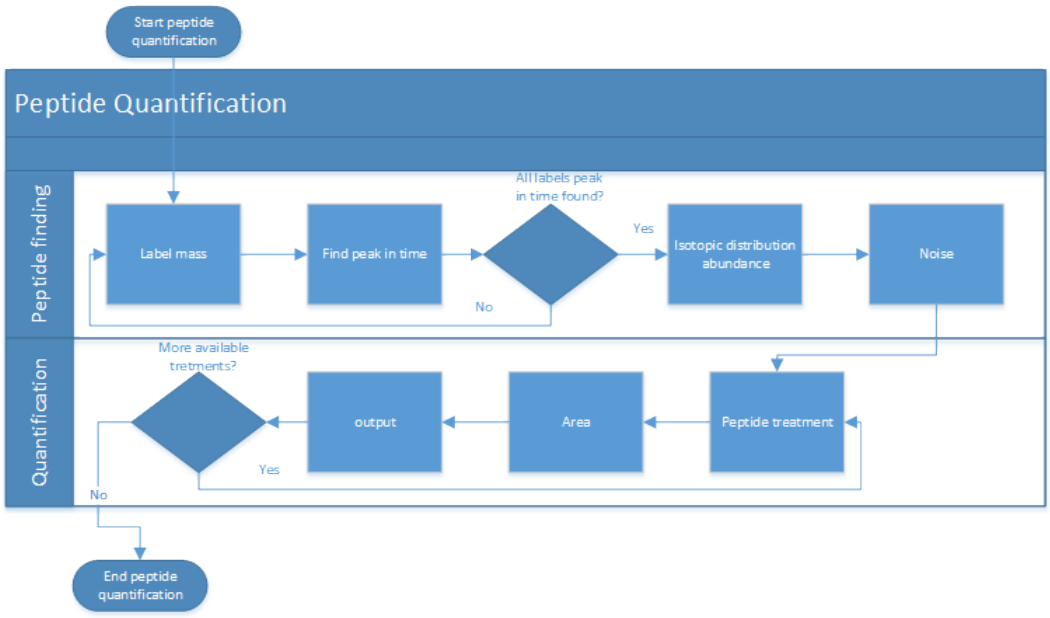


Figure 9: Block diagram showing the peptide quantification steps.

The figure 9 shows the peptide quantification process, where for every avail-

able label calculates the peptide mass and finds its peak in time. Once all the label's peak in time are found, there is an isotopic distribution abundance process (checks for intensity overlappings) and noise calculation. These steps closes what in this memory is cataloged as peptide finding. Following these steps, for every available peptide treatment: the treatment is done, the area is calculated, and the quantification output is stored. These steps are the previously mentioned quantification process.

In the next subsections all the mentioned processes are clearly explained and described in order to provide a better understanding of the algorithm.

3.1.1 Labels

As mentioned in a previous chapter, a sample is introduced to the mass spectrometer and it produces an output containing mz vs $intensity$ relations. This sample, at the same time, can be also formed by a serie of samples. If it is the case, this serie of samples are labeled. Every label is characterized by having a set of amino acids modified, this is having different masses. For example, let's have two different labels: Light and Heavy, where the first one does not have any amino acid modified, but the Heavy has the Lysine (K) mass modified, having an increment of +8 on its mass. This means that all peptides that contain the Lysine amino acid, will have an increment of +8 on their mass for every appearence of Lysine in their sequence. So that, the Light peptides will have a different mass than the Heavy ones.

Usually these amino acid modifications can be distinguished in an amino acid sequence because the modified amino acid is followed by a symbol: #, $, *, %, and so on.

Static masses Some amino acids can be introduced chemically altered with a mass modification. This means, that every time that the modified amino acid is found, the alteration mass must be summed.
In this case, the amino acid will not be followed by a symbol because it will be always found altered.

For example: let's consider the Cysteine (C) amino acid altered with a mass increment of +57. This means that for every appearence of Cysteine in a peptide sequence, +57 will be added to its mass.

Dynamic masses In this case, some amino acids can be chemically altered modifying its mass in some cases. This means, that sometimes the amino acid will be found with its original mass but other times will be found with its altered mass. For this reason, the amino acid modification will be marked using a symbol.

For example: let's consider the Methionine (M) amino acid altered with a mass increment of +16 marked with the * symbol. This means that for every appearence of the marked Methionine (M*) in a peptide sequence, +16 will be summed to its mass. If Methionine is found without the symbol in the peptide sequence, the mass modification will not be used.

Some dynamic modifications forbids more amino acid mass alterations. This means that if an amino acid is dynamically altered this way and a label has this same amino acid modified, the mass will not be summed.

Label masses As mentioned before, in a mass spectrometer run can be introduced more than one sample, where every sample is labelled. Every label can have a set of altered amino acids which have the mass altered and its annotated by a symbol.

In case that a label has the same amino acid altered that a dynamic modification that forbids its posterior alteration, two different scenarios can occur: if the peptide is found with the amino acid dynamic mark, all labels will use the same mass for that amino acid. The other scenario is that the amino acid is found without the mark, in this case, every label will add the corresponding mass modifications, so that the amino acid masses will be different.

The N-Terminus and C-Terminus are special amino acids that refer to the peptide start and end respectively. This means that if any of these amino acid is altered (statically, dynamically, or in a label) the mass alteration will be added only once in every peptide, due that is the peptide start or end.

The mass increment algorithm is very simple, for every marked amino acid counts the number of times that appears inside the peptide sequence and summes its mass alteration every time. If the marked amino acid is a N/C-Terminus, the mass will only be summed once.

Function getMassIncrement(PS, M)

Input: PS: peptide sequence, M: marked amino acids
Output: Additional mass for the given peptide
additional_mass $\leftarrow 0$;
for *every aminoacid in M* **do**
    mass_alt $\leftarrow$ *aminoacid* mass alteration;
    if *aminoacid is not Terminus* **then**
        appearences $\leftarrow$ *aminoacid* appearences in PS;
        additional_mass $\leftarrow$ *additional_mass* + *appearences* $*$ *mass_alt*;
    else
        additional_mass $\leftarrow$ *additional_mass* + *mass_alt*;
    end
end
return *additional_mass*

The peptide quantification starts setting the peptide's labels. The first thing that will be done is the peptide base mass calculation, this is the common mass that all labels will have (the amino acid sequence mass without modifications summed to the additional mass of the static and dynamic amino acid modifications found). Finally, for every label, the mass will be calculated summing to the base mass the additional mass of the marked amino acids that can be modifiable.

Procedure setPeptideLabels(PS, charge, L, SM, DM)

Input: PS: peptide sequence, charge: peptide charge, L: labels, SM: static marks, DM: dynamic marks

Result: All peptide's labels correctly set

$PeptideLabels \leftarrow []$;

$base_mass \leftarrow getMass(sequence) + getMassIncrement(PS, SM) + getMassIncrement(PS, DM)$;

$PS \leftarrow deleteNonModifiableAminoAcids(PS, DM)$;

for *every label in* L **do**

  $label_marks \leftarrow getLabelMarks(label)$;

  $label_mass \leftarrow base_mass + getMassIncrement(PS, label_marks)$;

  $label_mz \leftarrow massToMz(mass, charge)$;

  $PeptideLabels.push(newPeptideLabel(label, label_mz))$;

end

Check that in setPeptideLabels function, the non modifiable amino acids are deleted from the peptide sequence. This is done in order to prevent a mass alteration on amino acids dynamically modified that forbids more mass alterations.

The algorithm's next step is find for every label its peak along the time.

3.1.2 Peak in time finding

Once the peptide's label masses have been calculated, is time to find their peaks along the time. This is, search along the time the labeled peptide mz (mass to charge) inside the mass spectrometer output file and get their intensities. Sorting the intensities by retention time, a elution chromatogram is obtained; this chromatogram will be used a posteriori in order to calculate the labeled peptide area.

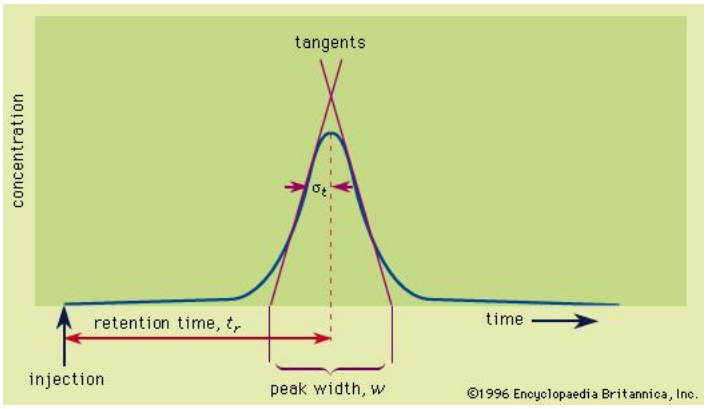


Figure 10: Elution chromatogram.

To do so, the algorithm needs the MS2 scan where the mass spectrometer precursored (detected) the peptide. From this scan, the algorithm moves along the retention time, accessing the previous and posterior MS1 scans where the labeled peptide mz was found and obtaining the corresponding intensities (peaks).

As the mass spectrometer outputs some noise peaks, in order to be sure that the get peaks are not noise, the algorithm searches their spectra. The number of spectral peaks to be found can vary, letting the user decide the number of peaks to be found in order to not consider the peak noise.

Peaks in time

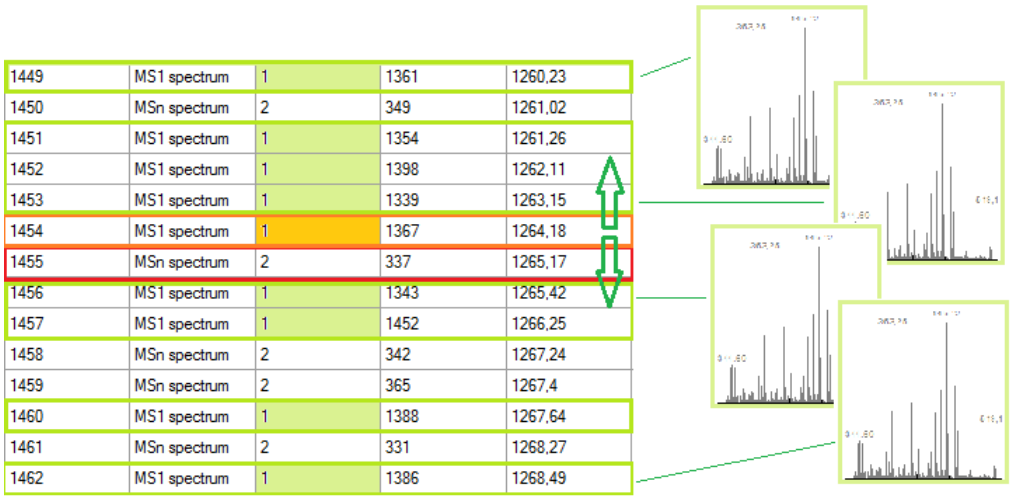
As mentioned before, the algorithm does not search only the peptide's peak along the time, also search its spectra on it in order to verify the peak correctness. The spectral search can be done in two different ways: defining a limit for the number of scans to be searched, or not. In any case, the search in the previous or posterior scans will stop if the spectra is not found in x consecutive scans.

Every spectral isotope has its own mass, and for this reason a different mz. As explained in a previous chapter, the spectra is a distribution of isotopes where every isotope has $(100 - i)$ Carbon 12 atoms $(12C)$ and i Carbon 13 atoms $(13C)$, where i is the isotope position in the distribution, starting by 0. This means, that every consecutive isotope of the spectra will have a mass

difference of $13C-12C$ with its correlatives. So that, at the time of searching the spectra inside scan, a set of different mz will be searched.

The peaks along the time search works as follows:

The algorithm calculates the mz (mass to search) for every spectral isotope of the peptide in order to be able to find the spectra within a scan. The function setSpectrasInTime moves from the given MS1 scan to the previous and posterior MS1 scans searching for the spectra. If it is not found, increments the number of consecutive scans that the spectra was not found; if this number is the maximum allowed, the search stops. Moreover, the algorithm has into account that if a window of scans where to look for had been set, and this maximum number is reached, the search stops too.



1449	MS1 spectrum	1	1361	1260,23
1450	MSn spectrum	2	349	1261,02
1451	MS1 spectrum	1	1354	1261,26
1452	MS1 spectrum	1	1398	1262,11
1453	MS1 spectrum	1	1339	1263,15
1454	MS1 spectrum	1	1367	1264,18
1455	MSn spectrum	2	337	1265,17
1456	MS1 spectrum	1	1343	1265,42
1457	MS1 spectrum	1	1452	1266,25
1458	MSn spectrum	2	342	1267,24
1459	MSn spectrum	2	365	1267,4
1460	MS1 spectrum	1	1388	1267,64
1461	MSn spectrum	2	331	1268,27
1462	MS1 spectrum	1	1386	1268,49

Figure 11: Peaks in time finding: The marked red scan is the precursor MS2 scan, the orange one is the MS1 referenced by the MS2. From this scan, the algorithm searches the spectra within the previous and posterior MS1 scans. Note that every MS1 scan contains a lot of ions to compare with. Columns: scan number, spectrum type, MS level, data points, scan time.

Spectra finding

This section will explain how the spectra is searched inside a MS1 scan. For every spectral isotope that must be found, its peak (mz vs $intensity$) will

Procedure setSpectrasInTime(MS1)

```
Input: MS1: nearest MS1 scan to the precursor MS2
Result: Spectras along the time
spectra_mzs ← calcSpectraMzs();
addSpectra(MS1, spectra_mzs);
maxNotFound ← getMaxScansNotFound();
notfound ← 0;
auxMS1 ← MS1;
scansLimit ← ∞;
i ← 0;
if defined scan window then
    scansLimit ← getScansLimit();
end
while i < scansLimit and notfound < maxNotFound do
    auxMS1 ← getPreviousMS1(auxMS1);
    if isValid(auxMS1) then
        spectra = addSpectra(auxMS1, spectra_mzs);
        if spectra not found then
            notfound ← notfound + 1;
        else
            notfound ← 0;
        end
    end
    i ← i + 1;
end
auxMS1 ← MS1;
i ← 0;
while i < scansLimit and notfound < maxNotFound do
    auxMS1 ← getPosteriorMS1(auxMS1);
    if isValid(auxMS1) then
        spectra = addSpectra(auxMS1, spectra_mzs);
        if spectra not found then
            notfound ← notfound + 1;
        else
            notfound ← 0;
        end
    end
    i ← i + 1;
end
```

be searched within the scan. This is done looking for a mz coincidence. At this point, must be considered that the mass spectrometer outputs the data with some error associated in parts per million units (ppm). This error can be defined in the configuration file by the user, otherwise the default value ($5ppm$) will be used. This way, the isotope will be searched in a mz window of $mz \pm ppm$. But, which mz will be used? There are two different possibilities:

- Theoretical mz: is the previously calculated mz.
- Variable mz: following the theory that adjacent scans have a similar ppm variation associated, the mz window will be defined using the mz found in the previous scan.

This means that if the Theoretical mz option is selected, the searched mz window will always be $theoretical_mz \pm ppm$, while in the other case will depend on the mz that was found in the previous scan. Lets consider that the algorithm is searching for a mz of 12.8 and the possible error is 0.05; in the actual scan the isotope is found with a mz of 12.79, so that in the next scan the isotope will be searched within a 12.79 ± 0.05 window.

Sometimes, the algorithm finds more than one possible peak (isotope) inside the mz window, if it happens, the algorithm can do two different things:

- Select the one with the most similar intensity to its previous scan. (Remember that the isotopes sorted produce a peak shape using their intensity)
- Select the one with the most similar mz to its previous scan if the variable mz was selected, or the one with the most similar mz to the theoretical one, otherwise.

The explained getSpectra function is written below:

```
Function getSpectra(scan, spectraMzs)
    Input: scan: MS1 scan number, spectraMzs: spectral isotopes mzs
    Output: The found spectral peaks
    spectra ← [];
    possiblePeaks ← [];
    for mz in spectraMzs do
        if variable mz option then
            mz ← getPreviousMz();
        end
        possiblePeaks ← getPossiblePeaks(scan, mz);
        peak ← Ø;
        if similar intensity option then
            peak ← getNearestIntensity(possiblePeaks);
        else
            peak ← getNearestMz(possiblePeaks, mz);
        end
        if peak valid then
            spectra.push(peak);
        end
    end
    return spectra;
```

The *getPossiblePeaks* function is a trivial search inside a sorted set of peaks. This set is sorted by mz so that the search is more efficient.

At this point, the algorithm has all the Label's peaks, so that an elution chromatogram can be obtained, in order to calculate their area.

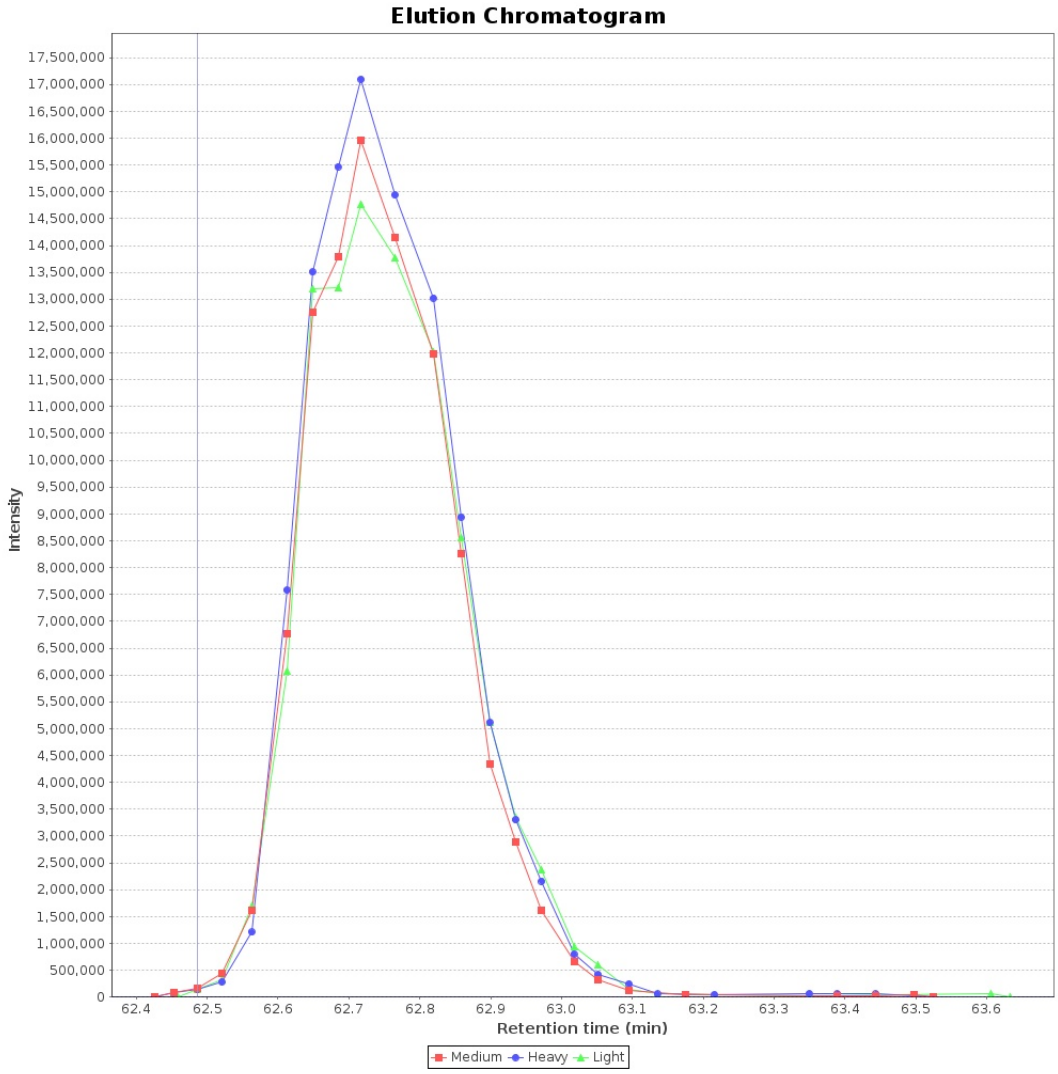


Figure 12: Elution chromatogram of three different labels: Light (green), Medium (red), Heavy (blue).

3.1.3 Isotopic distribution abundance

Most elements occur in nature as a mixture of isotopes. Isotopes are atom species of the same chemical element that have different masses. They have the same number of protons and electrons, but a different number of neutrons. The main elements occurring in proteins are CHNOPS. The nucleon

number is denoted in the upper left corner of an atom, such as 12C for the carbon 12 isotope. Several isotopes of an element can be found in nature and are called natural isotopes. The natural isotope with the lowest mass is called monoisotopic.

Here is a list of the heavy isotopes, sorted by abundance. It is important to notice that abundances of natural isotopes are no physical constants: These abundances vary depending on time and place (continent, planet, solar system) where the sample is taken. Therefore, it can be used in order to know its origin.

Isotope	Mass [Da]	%Abundance
34S	33.967868	4.21
13C	13.003355	1.10
33S	32.971459	0.75
15N	15.000109	0.37
18O	17.999159	0.20
17O	16.999131	0.038
2H	2.014102	0.015

Table 2: List of the most heavy isotopes sorted by abundance.

At the above table can be seen that sulfur has a big impact on the isotope distribution. But it is not always present in a peptide (only the amino acids Cystein or Methionin contain sulfur). Apart from that, 13C is the most abundant. As the "backbone" of all biomolecules is made from carbon, often the elements are classified based on their similarity or dissimilarity to carbon and, for this reason, carbon is used in order to identify the isotope distribution.

The isotopic distribution abundance can be understood as a distribution of peaks that has $(100 - i)$ Carbon 12 and i Carbon 13, where i is the isotope position in the distribution starting by 0.

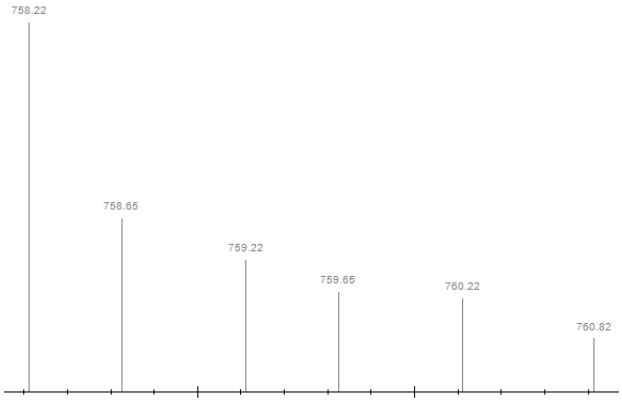


Figure 13: Isotopic distribution abundance (mz vs $intensity$): Monoisotopic, first isotope, second, third, forth and fifth.

In some cases, when different labels are used in a mass spectrometer run, the isotopic distribution abundance of one label can be overlapping the isotopic distribution abundance of another label. If this happens, the algorithm should be able to identify the "real" intensity of a given label peak.

In figure 14, an example is shown for a better understanding. In this case there are three different labels: light (blue), medium (red) and heavy (yellow). Can be observed that the 4th and 5th isotopes are overlapping the monoisotopic and the 1st isotope, and therefore the intensities are wrongly assigned. For example, the medium's monoisotopic will have the same intensity than the light's 4th isotope, when the light's 4th isotope should have a lower intensity.

In order to solve the overlapping problem, the algorithm will use a subalgorithm that will recalculate the peak intensity for a given label. For doing so, the algorithm will need to calculate for every label the theoretical isotopes distribution abundance and subtract from the real distribution the proportional intensity of the other overlapped label's isotope.

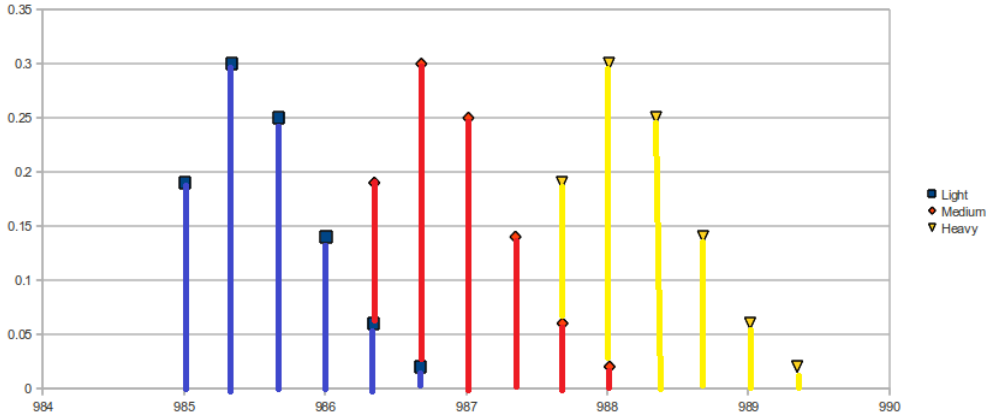


Figure 14: Overlapped isotopic distribution abundance of three labels: Light (blue), Medium (red), and Heavy (yellow).

In order to calculate the theoretical isotopic distribution abundance, a function called *calcSpectra*[1] will be used. With this function, the non-labeled (Light) distribution will be calculated using a charge of 1. This way, mass will be equals to mz and will be easier to calculate the other labels masses.

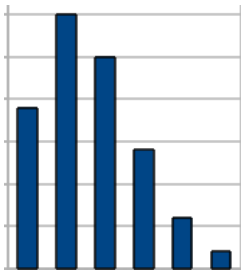


Figure 15: Theoretical isotope distribution abundance.

In the figure 15 is shown the Light isotope distribution abundance calculated by the mentioned function. Comparing this image to the figure 14 can be seen that follows exactly the same distribution than its Light distribution.

[1]Function programmed by Daniel Hernandez located in the *edu.uw.villenlab.Common* library.

As only the Light distribution has been calculated, and was calculated without having into account the amino acid modifications, all the other isotope distributions,including the Light one, will be calculated using proportional intensities to the provided distribution. The algorithm will work as follows:

Function getTheoreticalIsotopicDistribution(PS, charge, L, DM, SM)

Input: PS: peptide's sequence, charge: peptide's charge, L: peptide's labels, DM: dynamic modifications, SM: static modifications

Output: All the peptide's isotopic distribution abundances (a distribution for every label)

$thDist \leftarrow []$;
$thDist0z \leftarrow calcSpectra(PS).getDistribution()$;
$pepAdditionalMass \leftarrow$
$getMassIncrement(PS, SM) + getMassIncrement(PS, DM)$;
$monoisotopicIntensity \leftarrow getMonoisotopicIntensity(thDist0z)$;
for *every label in L* **do**
  $label_marks \leftarrow getLabelMarks(label)$;
  $labelAdditionalMass \leftarrow$
  $pepAdditionalMass + getMassIncrement(PS, label_marks)$;
  $labelPeaks \leftarrow []$;
  for *peak in thDist0z* **do**
    $iso_mz =$
    $getMz(peak) + massToMz(labelAdditionalMass, charge)$;
    $iso_propInt = getIntensity(peak)/monoisotopicIntensity$;
    $iso_peak \leftarrow Peak(iso_mz, iso_propInt)$;
    $lanelPeaks.push(iso_peak)$;
  end
  $thDist.push(labelPeaks)$;
end
return *thDist*;

The *calcSpectra* function provides a spectra of accumulative intensities relative to the monoisotopic one. This spectra has for every isotope the mass calculated using a charge of one, meaning that mass is equal to mz. For this reason, for every label and isotope, the real mz must be recalculated (adding the amino acid modifications and using the real charge in order to obtain

the real theoretical mz). Once the mz has been calculated, the algorithm calculates the isotope percentage intensity relative to the monoisotopic one. This isotope's mz vs *intensity* is stored for every label's isotope in order to recalculate their theoretical intensity.

Once the theoretical distribution for every label is obtained, the overlapped isotopes can be found and their intensities recalculated.

```
Procedure checkIsotopeDistributionOverlapping
    Input: labels: peptide's labels
    Result: Intensities recalculated for the overlapped isotopes
    labelsThDist ← getTheoreticalIsotopeDistribution();
    for i = 1 to i = labels.size do
        label ← labels.get(i);
        prevLabel ← labels.get(i − 1);
        labelDist ← labelsThDist.get(i);
        prevLabelDist ← labelsThDist.get(i − 1);
        for isotope in labelDist do
            isoLowMz ← getLowMz(isotope.getMz());
            isoHighMz ← getHighMz(isotope.getMz());
            for prevIsotope in prevLabelDist do
                prevIsoLowMz ← getLowMz(prevIsotope.getMz());
                prevIsoHighMz ← getHighMz(prevIsotope.getMz());
                if isoLowMz or isoHighMz contained between prevIsoLowMz
                and prevIsoHighMz then
                    isotopeRecalculation(label, isotope, prevLabel,
                    prevIsotope);
                end
            end
        end
    end
```

In the checkIsotopeDistributionOverlapping procedure can be seen that for every label and isotope, its mz is compared to every isotope's mz of the previous label (label with lower mass). This comparison is done having into account the possible mass spectrometer precision loss on the mz output, and

for this reason, the *ppm* window previously used at the peak searching will be used. To consider an isotope possibly overlapped, an mz inside its mz window must be contained inside the other isotope's mz window. If it is the case, these two isotope peaks will be recalculated. To do so, will be necessary to obtain the monoisotopic intensities and multiply each one by their isotope's relative percentage. Remember that the previous calculated intensities are intensity percentages relative to their monoisotopic.

The recalculation is done as follows:

Procedure isotopeRecalculation(label, isotope, prevLabel, prevIsotope)

Input: label: Label that has the isotope overlapped, isotope: label's overlapped isotope, prevLabel: the previous label that is overlapped to *label*, prevIsotope: the prevLabel's overlapped isotope

Result: Overlapped isotopes' intensity recalculated at the whole retention time

$mz \leftarrow isotope.getMz()$;
$prevMz \leftarrow prevIsotope.getMz()$;
for *retT in all retention time* **do**
  $iso \leftarrow label.getSpectra(retT).getIsotope(mz)$;
  $prevIso \leftarrow prevLabel.getSpectra(retT).getIsotope(prevMz)$;
  if *iso is prevIso* **then**
    $monoInt \leftarrow$
    $label.getSpectra(retT).getMonoisotopicIntensity()$;
    $prevMonoInt \leftarrow$
    $prevLabel.getSpectra(retT).getMonoisotopicIntensity()$;
    $intensity \leftarrow monoInt * isotope.getIntensityRelPerc()$;
    $prevIntensity \leftarrow$
    $prevMonoInt * prevIsotope.getIntensityRelPerc()$;
    $percentage \leftarrow intensity/(intensity + prevIntensity)$;
    $intensity \leftarrow iso.getIntensity() * percentage$;
    $prevIntensity \leftarrow prevIso.getIntensity() * (1 - percentage)$;
    $iso.setIntensity(intensity)$;
    $prevIso.setIntensity(prevIntensity)$;
  end
end

For every retention time where those labels' isotopes were found will be checked if the isotope is exactly the same than the other isotope. If it is the case, the isotopes' intensities will be calculated using the known intensity percentages relative to the monoisotopic. Moreover, these intensities won't be enough accurate because they are not using the mass spectrometer intensity output. For this reason, knowing the new calculated intensities, the real

intensity percentage for these two isotopes will be calculated allowing the real intensity calculation.

After this sub-algorithm execution all the overlapped isotopes' intensities will be recalculated correctly.

Visual example
The following example is centered on the monoisotopic isotopes overlapping, i.e. the spectra's first isotope.

Let's suppose that there are three different labels: Light, Medium and Heavy, and the mass spectrometer has found the next isotopes' mz vs intensity.

Isotope	mz	intensity
5th isotope	986.20	593833.5

Table 3: Overlapped Light isotope

Isotope	mz	intensity
1st isotope (monoisotopic)	986.20	593833.5
5th isotope	987.53	645206.25

Table 4: Overlapped Medium isotopes

Isotope	mz	intensity
1st isotope (monoisotopic)	987.53	645206.25

Table 5: Overlapped Heavy isotope

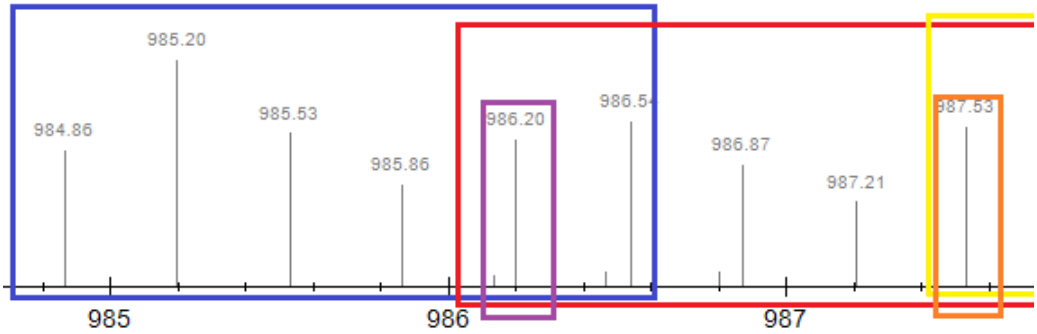


Figure 16: Light (blue), Medium (red) and Heavy (yellow) spectras. In purple is squared the Light-Medium isotope overlapping, in orange the Medium-Heavy overlapping.

The image 16 shows two monoisotopic isotope overlappings: the Medium monoisotopic, and the Heavy one. The Medium monoisotopic is overlapped with the 5th Light's isotope, so both of them were output (by the mass spectrometer) with an mz of 986.20 and an intensity of 593833.5. This intensity is the result of the both isotopes' intensity summed. The same happens with the Heavy's monoisotopic, it is overlapped with the Medium's 5th isotope, and both of them have an mz of 987.53 and an intensity of 645206.25. Note that the Heavy's 6th isotope is overlapped with the Medium's 2nd isotope too, but in this example only the monoisotopic overlappings will be considered.

Following the explained algorithm, the first step is to obtain for every label their theoretical distribution abundance. The next step is to detect the possible overlapped isotopes. The first overlapping detected will be the Medium's monoisotopic with the Light's 5th isotope. So, after this detection, their intensities will be recalculated in every retention time where these two isotopes were found. The same will happen with the other overlappings. After this recalculation the isotopes' mz vs *intensity* will be as follows:

Isotope	mz	intensity
5th isotope	986.20	134701.89

Table 6: Overlapped Light isotope with its intensity recalculated

Isotope	mz	intensity
1st isotope (monoisotopic)	986.20	459131.6
5th isotope	987.53	118385.64

Table 7: Overlapped Medium isotopes with their intensities recalculated

Isotope	mz	intensity
1st isotope (monoisotopic)	987.53	526820.6

Table 8: Overlapped Heavy isotope with its intensity recalculated

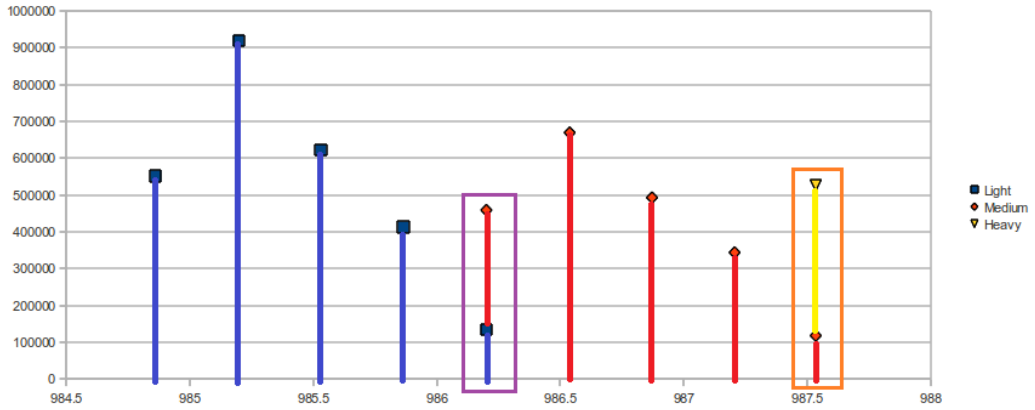


Figure 17: Recalculated labels' distribution abundance. Light (blue), Medium (red), Heavy (yellow).

In figure 17 can be seen how the monoisotopics "real" intensities were correctly recalculated.

3.1.4 Noise

The paper *The Impact of Peptide Abundance and Dynamic Range on Stable-Isotope-Based Quantitative Proteomic Analyses - Journal of Proteome Research*[1] explains that noise peaks are different that peptide peaks; noise peaks are more abundant and have lower intensities than peptide peaks. These differences can be used in order to find the peptide noise. The authors stated that:

"To determine chromatographic peak S/N, we first developed

> a method for background noise estimation in full scan (MS) spectra. Noise may arise from a number of factors including atmospheric sources, electrical interference, and chemical contaminants. These sources combine to produce noise peaks, which are observed alongside peaks generated from the detection of peptide ions. However, the intensities of most noise peaks are characteristically different from peptide signal: noise peak intensities are typically lower and more similar than those from peptide signal. The intensity distribution of all peaks measured within several m/z and retention time windows of varying sizes indicated that many noise peaks reproducibly fell within a narrow intensity range, and each window had a similar median intensity. We defined this median intensity observed in the local vicinity of the chromatographic peak as the noise level for this study, providing a suitable estimation of the background noise. The S/N of a particular peptide's chromatographic peak can then be calculated by dividing the maximum peak intensity by the noise level."

The peptide noise will be calculated searching all the peaks within a mz window inside a scan window. This means that the algorithm will have a set of scans where to look for peaks that have a mz contained in a concrete mz window. This mz window will be defined using the peptide's highest mz for the higher part of the window, and the lowest mz for the lower part. This means that the window will go from the $lowest_mz - mzNoiseWindow$ to the $highest_mz + mzNoiseWindow$. And the peaks will be searched in a scan window that has been defined using the peptide's precursor scan as the window's central scan. This is $precursorScan \pm msNoiseWindow$. Finally, these peaks' intensities will be sorted, and the median obtained will be the peptide's noise.

As the noise peaks are more abundant and have lower intensities, the median of the different intensity ranges sorted by frequency will fall at the noise range.

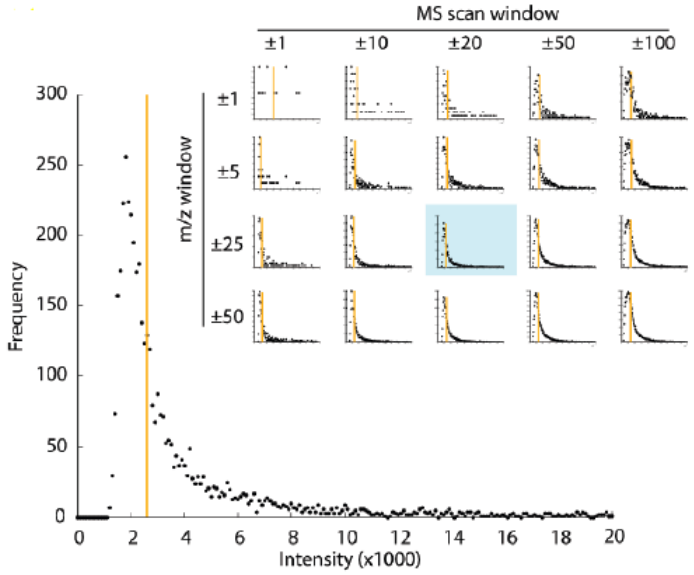


Figure 18: *Image showing the noise localization using the median of the different intensity ranges sorted by frequency. This image shows how changing the set windows the median always falls at a similar position.*

In the figure 18 can be appreciated that varying the defined windows, the noise always falls at a similar position or range.

The algorithm is described in the calcNoise function:

Function calcNoise(lowMz, highMz)

Input: lowMz: peptide's lowest mz, highMz: peptide's highest mz
Output: noise: peptide's noise
$minMz \leftarrow lowMz - mzNoiseWindow$;
$maxMz \leftarrow highMz + mzNoiseWindow$;
$intensities \leftarrow []$;
for *every scan in msNoiseWindow* **do**
    $scanInt \leftarrow scan.getIntensitiesBetweenMz(minMz, maxMz)$;
    $intensities.push(scanInt)$;
end
$intensities \leftarrow sort(intensities)$;
$noise \leftarrow getMedian(intensities)$;
return *noise*;

3.1.5 Peptide treatment

At this point the algorithm has obtained the peptide's peak in time for every available label. Moreover, the algorithm has recalculated the overlapped isotopes and calculated the peptide's noise. With that the algorithm has enough information in order to proceed to quantify, but for a better accuracy the algorithm does some peptide treatments.
Usually the peak in time does not have a 'perfect' peak in time shape, this means that does not have a perfect gaussian bell shape. The problem is that some of these peaks have some error associated related to the mass spectrometer. For this reason, the algorithm does some peptide treatments in order to obtain a better accuracy when the quantification is performed.

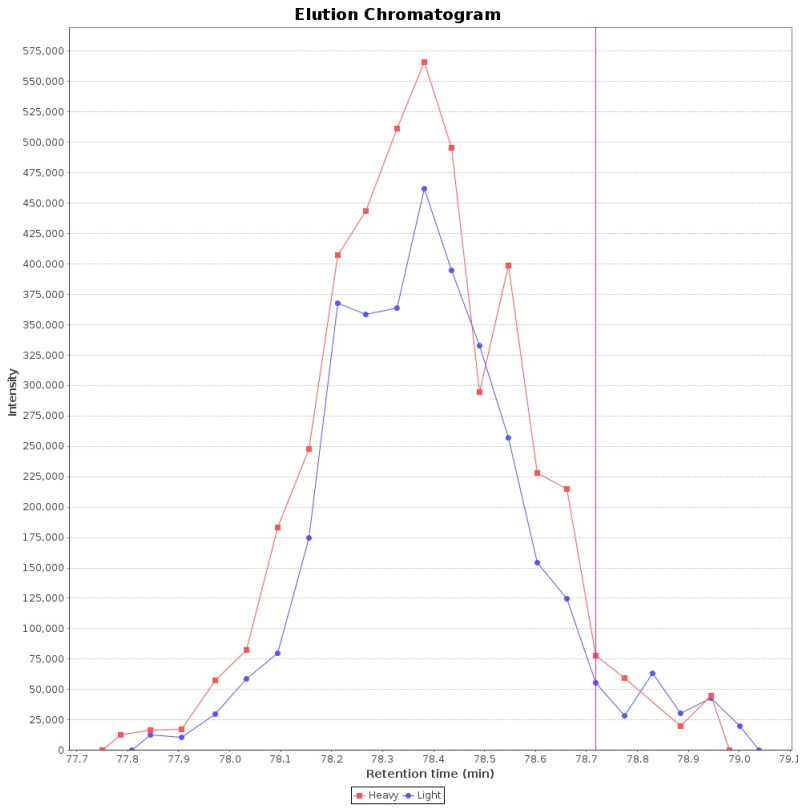


Figure 19: Heavy and Light peak in time

There are three different methodologies related to the peptide's treatment: smooth, reconstruct and calculate the PPM Score. These methodologies will be detailed in the next subsections.

On the other hand, the algorithm has a quality score that determines how good is the peptide's shape. The quality score consists in calculate how many peak shapes can be found and divide it by the total number of peaks that compounds the whole peptide shape. Of course, the algorithm should subtract one peak from the total number of peak shapes because in the elution chromatogram will always be, at least, the main peak. This score will be multiplied by 100 in order to obtain the percentage of peaks; as higher is the score, worst is the peptide's quality.

A peak will be found if the previous and the posterior intensities are less

than the actual one.

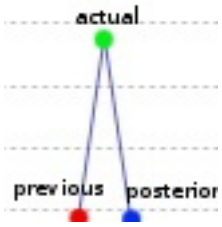


Figure 20: Peak shape.

Here can be seen how the algorithm works:

Function qualityScore

Input: intensities: peak shape intensities
Output: score: quality score
$int1 \leftarrow intensities.get(0)$;
$int2 \leftarrow intensities.get(1)$;
$int3 \leftarrow 0$;
$score \leftarrow 0$;
for $i \leftarrow 2$ *to intensities.size()* **do**
  $int3 \leftarrow intensities.get(i)$;
  if *(i1 <i2) and (i3 <i2)* **then**
    $score \leftarrow score + 1$;
  end
  $int1 \leftarrow int2$;
  $int2 \leftarrow int3$;
end
$score \leftarrow ((score - 1)/intensities.size()) * 100$;
return *score*;

Smooth
The smoothing algorithm is simple, it will calculate for every point in the retention time, a new intensity that is the mean between its actual intensity and the previous and posterior ones. It is important to have into account that the first and the last point must always be intensities zero, if aren't, the peak could become a different shape (as shown in the below image).

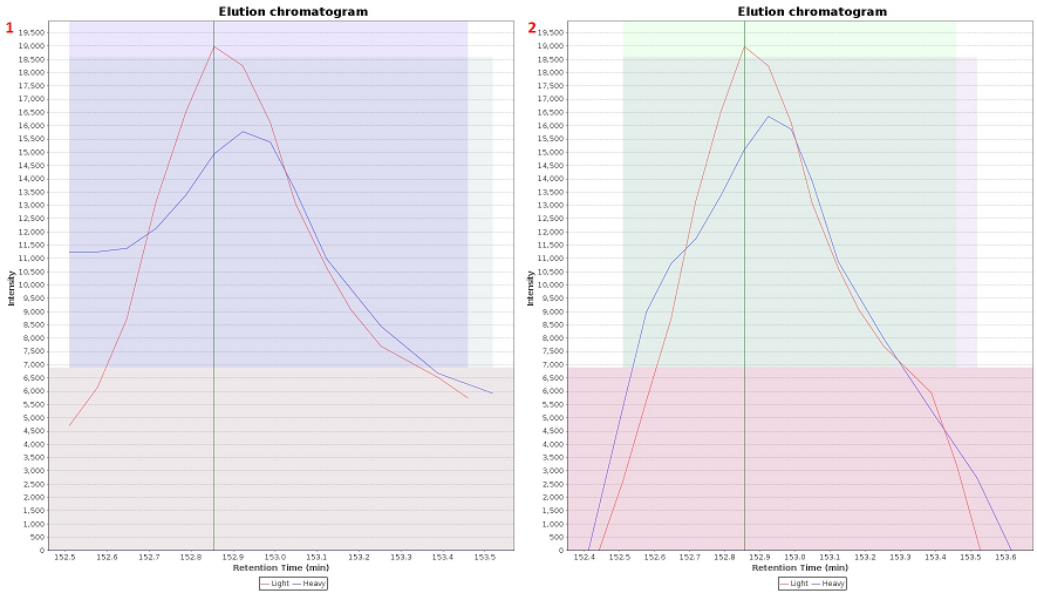


Figure 21: The first image (1) is the resulting peak in time shape that become after smoothing the peptide without having into account the start and end 0 intensities. The right image (2) is the resulting shape having into account the 0 intensities.

As can be seen in figure 21, the smoothing without the 0 intensities wasn't working as expected, the shape changed because they hadn't stable points. In the other hand, applying zero intensities the algorithm obtained the expected peak in time.

But doing so, another problem has appeared. As can be seen in a below image, if the whole peak in time only have a few points (for example three or four points), the peak decreases a lot if the zero intensities are maintained. This means that in some cases the highest peak in time could become the lowest one after the smoothing. To solve this problem, the algorithm smooths the first and last intensities too. Once the smoothing is finished, the algorithm will change the first and last intensities to zero. This way the ratios between the peptide's labels are maintained.

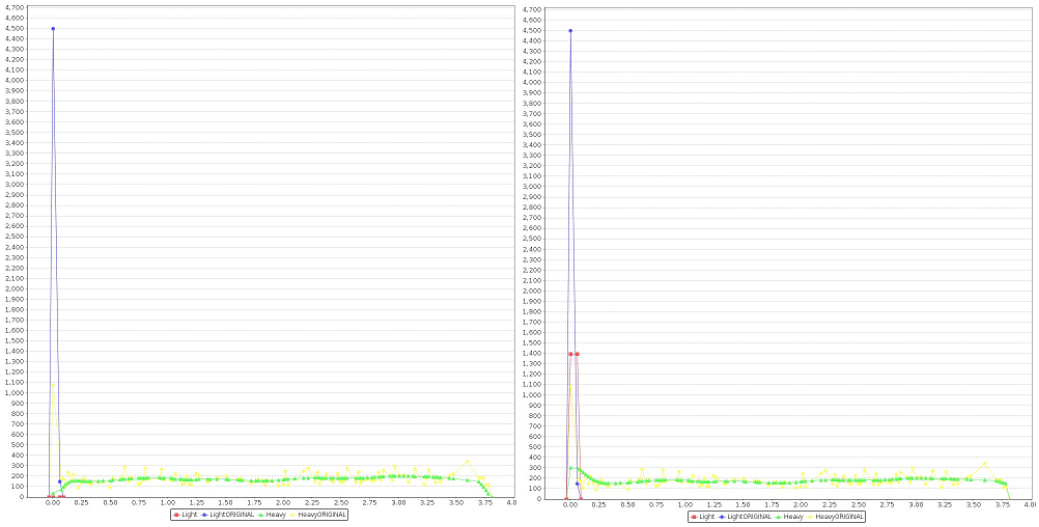


Figure 22: These two images show the light (yellow and green) and heavy (blue and red) peak in time shapes before (heavy: blue, light: yellow) and after (heavy: red, light: green) the smoothing. The first image (left) shows the shape obtained maintaining the zero intensities all the time. The second image (right) shows the shape obtained smoothing the zero intensities too. As can be seen, in the first image the heavy peak almost disappeared while the light one smoothed correctly the shape, in this case the light and heavy ratio changed completely, and for this reason this kind of smooth cannot be applied. In the other hand, the second image 'correctly' maintained the ratio.

The smoothing algorithm works as follows:

Procedure smoothPeptide

Input: intensities: peak in time intensities
Result: intensities: peak in time smoothed intensities
$int1 \leftarrow intensities.get(1)$;
$int2 \leftarrow intensities.get(2)$;
$int3 \leftarrow 0$;
$y \leftarrow (int1 + int2)/3$;
$auxInts \leftarrow []$;
$auxInts.push(0)$;
$auxInts.push(y)$;
for $i \leftarrow 3$ *to intensities.size() - 1* **do**
    $int3 \leftarrow intensities.get(i)$;
    $y \leftarrow (int1 + int2 + int3)/3$;
    $auxInts.push(y)$;
    $int1 \leftarrow int2$;
    $int2 \leftarrow int3$;
end
$y \leftarrow (int1 + int2)/3$;
$auxInts.push(y)$;
$auxInts.push(0)$;
$intensities \leftarrow auxInts$;

This algorithm puts as the first and last intensities a zero intensity, and calculates for the other peaks their smoothed intensities. These intensities are recalculated doing the average of the actual, previous and posterior intensities.

The maximum number of smooths that can be done for a peptide is determined by the quality score and/or provided by a parameter that says the maximum smooths that can be done.

It is important to note that every different label of the current peptide must be smoothed the same number of times in order to maintain the ratios. This is important because maybe a label needs less smooths in order to have a perfect quality score, and if the other labels need more smoothing their ratio

respect to the label that has stopped changes.
For this reason, the number of smooths that will be performed for every label will be the maximum. To determine this maximum, all labels will be smoothed until their own maximum will be reached. Once this maximum number of smoothings is obtained for every label, the algorithm will look which had the maximum number of smoothings. Then, every label will be smooth until this maximum number is reached.

Procedure smoothAllLabels

```
Input: labels: Peptide's labels
Result: Labels' smoothed peak in time
max ← 0;
for label in labels do
    PiT ← label peak in time;
    smooths < −PiT.maxSmooth();
    if smooths >max then
        max ← smooths;
    end
end
for label in labels do
    PiT ← label peak in time;
    PiT.smoothTimes(max);
end
```

In the figure 23 can be seen the difference between a peptide smoothed and the original shape that it has. In this image can be seen that the intensities change (decrease) but the ratio between labels is maintained. The ratio is the main information that the algorithm wants to obtain.

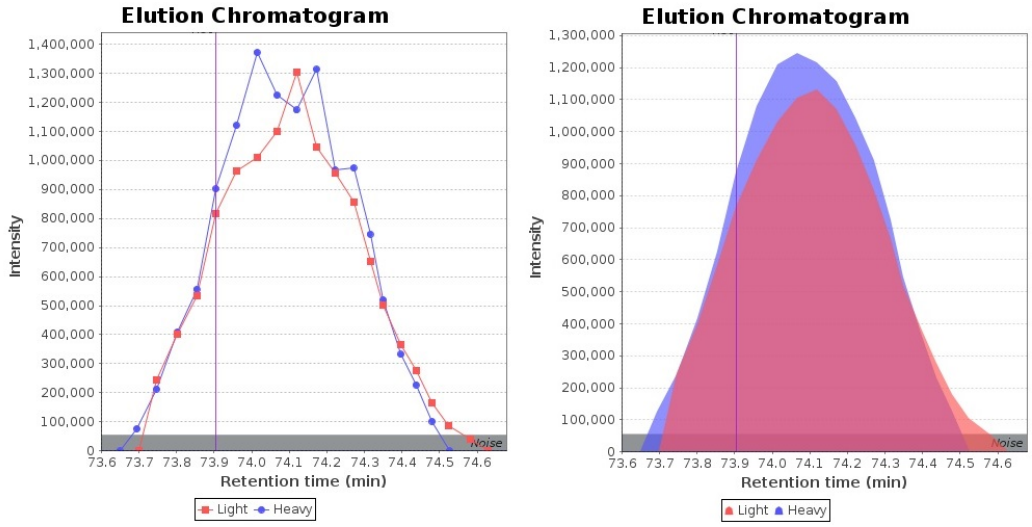


Figure 23: These images show how changed the elution chromatogram after the smoothing is performed. There are two labels: light (red) and heavy (blue). The left image is the original chromatogram and the right one is the elution chromatogram obtained after the smooth is done.

Smooth Limits

The smoothed peak in time can be used in order to obtain its limits (identify and delete the possible tail).

A normal smoothed peak in time shouldn't have more than one peak, but in some cases the peak in time obtained contains more than one peak (peptide). When this happens, the algorithm should be able to identify which of these peaks (peptide) is the expected one. To do so, some different algorithms were considered:

Algorithm 1: Using the minimum slope

The basic idea of this algorithm, is to obtain the minimum acceptable slope for the peak. The utility of this, is to search points where the slope is stabilized. These stabilized points are considered peak tops or peak tails, and for this reason a limit should be established.

In this algorithm the minimum slope that the slope between two points (peaks) should have will be the mean of the two contiguous slopes to the

MS1 scan position. The reason of doing so, is that the MS1 scan position determines which of the peptides is the one that should be taken into account in case of having more than one peptide inside the points obtained. For doing so, the algorithm has to consider the cases where the MS1 is the first scan or the last scan, in these cases will be used the two posterior or previous slopes respectively.

$$slope = (i2 - i1)/(r2 - r1) \tag{1}$$

The slope between two points is the division between their intensity difference and their retention time difference.

Once the minimum slope is obtained, the algorithm will move from the MS1 scan to the left and to the right in order to find the limits:

Moving to the left side:
First of all it is important to know if the current position is in an increasing or decreasing zone, for this reason the algorithm will compare the previous and the actual intensities. If the previous is greater, will be considered that while the algorithm is moving to the left, the values are increasing and for this reason can be said that the MS1 scan is at peak right side. If the values are decreasing, can be said that the MS1 is at the peak left side.
Now, the algorithm will be moving to the left until the start limit is found or there aren't more points. To find the start limit will be considered some conditions. If the actual slope is less than the minimum, there will be two options:

- If it is an increasing zone, will be considered that is the top of the peak and for this reason the algorithm can continue and will be established a permitting zone until it will start to decrease.
- Else if the algorithm is in a permitted zone, will continue searching.
- Otherwise if at least one valid point was found, this peak will be the start limit.

Moving to the right side:
As in the left side was done, the algorithm needs to know if it is an increasing or decreasing zone while moving to the right, so if the posterior intensity is

greater than the actual will be considered that the MS1 scan is in an increasing zone. The conditions needed to find the end limit are exactly the same than the start limit ones.

This algorithm didn't work well enough. The minimum slope wasn't enough accurate to find the limits. In algorithm 4 will be shown a new algorithm based on this one that uses a moving minimum slope.

Algorithm 2: Using angles

As algorithm 1 didn't work as expected, was decided to try a new methodology using the slope angle instead of itself. In this case was considered that an angle inferior than 25^{o} should be a limit or a peak top.

Doing so a problem was found: as bigger were the intensities, bigger were the differences and for that reason, the slopes. That implied that the angle wasn't working well enough because for some values the $atg(m)$ normalized the angle around 80. In order to use this methodology the intensities should be normalized.
At this point another problem was found: if the intensities are normalized, in chromatograms where appears two different peak shapes, one big and one small, the small one will have a really small angles and would be seen as limits instead of peaks. For this reason, the algorithm was discarded.

Algorithm 3: Derivatives

Was considered convenient to use derivatives in order to find the stabilized zones. This means that the lower values of the derivatives should be tails or peak tops because their slopes are clearly smaller than the increasing or decreasing zone of a peak shape.

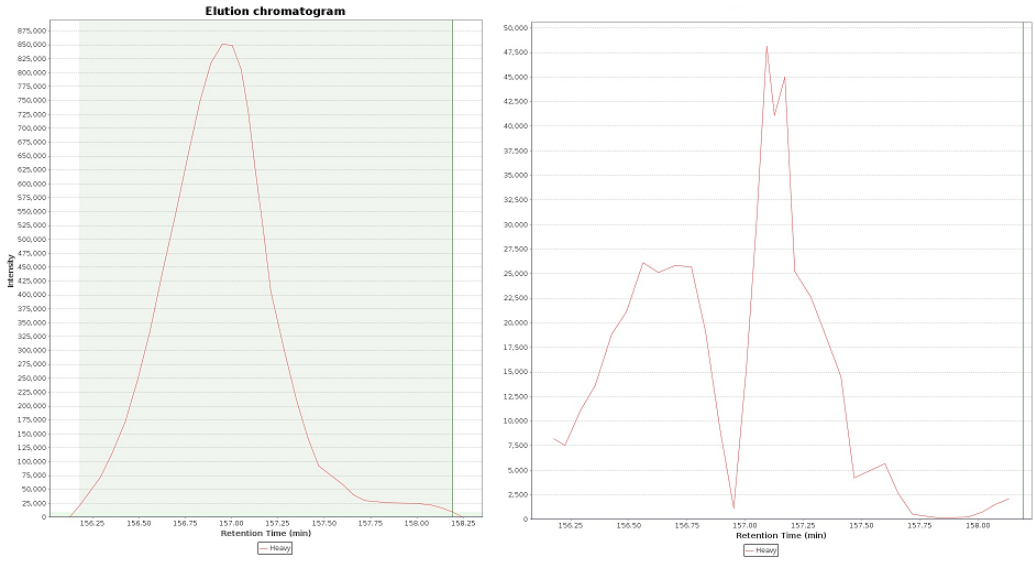


Figure 24: Peak in time (left) and its 1st derivative (right).

The problem here is to find this smaller values, which ones should be used? Is the same problem that is trying to be fixed: find limits.

To solve this, was tried to use the second derivative in order to find the inflection points (a point where the second derivative of a function changes the sign). As the smoothed peak is not a perfect peak (it has more inflection points), the algorithm didn't work as expected and wrongly assigned limits. The third derivative didn't work as well.

Algorithm 4: Moving slope

Finally, was decided to use the same idea as the first algorithm, but this time using a variable slope. This means that for every point, the minimum slope accepted for that peak (point) will be recalculated. For a better comprehension, the algorithm has been splitted (specialized) in two algorithms: for one peak shape in the elution chromatogram or more than one.

The first part of the algorithm will consist in check if there is only one peak or not. To do so, the only thing that will be done is check if there are more than one peak shapes (previous and posterior intensities lower than the actual one). This is the same algorithm that calculates the quality score

(function qualityScore).

More than one peak shapes:

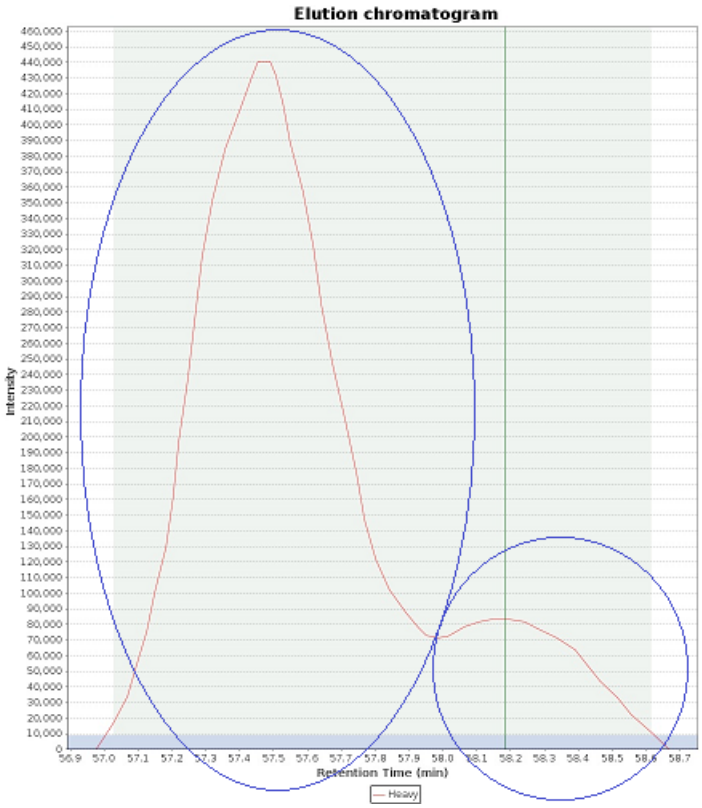


Figure 25: Smoothed peak in time where can be seen two different peak shapes (surrounded in blue circles).

In this case, starting from the MS1 scan position the algorithm will search the nearest top peak position. Is done this way, because this MS1 scan is the nearest one to the scan where the peptide was precursored, and for this reason is the most probable to be the correct shape. Once this top is found, the algorithm will look for the peak shape left and right limit.

To find the nearest peak, will be checked if the MS1 scan is in an increasing

```
if Is not one shape then
    intpos = findNearestPeakPos();
    limitL = findPeakLeftLimit(pos);
    limitR = findPeakRightLimit(pos);
```

or decreasing zone. If it is in an increasing zone, means that the MS1 scan is at the left side of the peak, and in this case the algorithm will move to the right until a point where the current intensity will be bigger than the posterior one (peak top). Otherwise, the MS1 scan is at the right side of the peak, so the same procedure will be done but this time moving to the left along the retention time.

```
Function findNearestPeakPos
    Input: intensities: peak in time intensities, ms1Pos: ms1 scan index
    Output: peak in time top peak index
    current ← intensities.get(ms1pos);
    next ← intensities.get(ms1pos + 1);
    ms1InLeft ← (current <next);
    if ms1InLeft then
        current ← next;
        for i ← ms1pos + 1 to intensities.size() - 1 do
            next ← intensities.get(i);
            if current ≥ next then
                return i;
            end
            current ← next;
        end
    else
        for i ← ms1pos decreasing to 1 do
            previous ← intensities.get(i − 1);
            if current ≥ previous then
                return i;
            end
            current ← previous;
        end
    end
    return -1;
```

Now that the top peak position is found, the algorithm can move along the right and left sides in order to find the limits. The algorithm in both cases are the same but moving to different sides. For this reason, only the left one will be explained.

Finding the left limit
First of all, the algorithm will calculate the slope that the top of the peak is forming with its left point. Once it is obtained, the algorithm will move from this point to the left calculating the slope and comparing it with the previous

one calculated. If the actual slope is less than the 30% of the previous slope, will be considered that the limit is reached. Less than a 30% means that it is a zone where is not clearly decreasing (stable or pre-stable zone). This zone should be the tail of the peak.

Function findPeakLeftLimit(startPos)

```
Input: startPos: peak in time top peak index
Output: peak in time left limit index
intPrev ← intensities.get(startPos − 1);
intAct ← intensities.get(startPos);
retTPrev ← retentionTime.get(startPos − 1);
retTAct ← retentionTime.get(startPos);
mPrev ← (intAct − intPrev)/(retTAct − retTPrev);
intAct ← intPrev;
retTAct ← retTPrev;
for i ← startPos − 1 decreasing to 1 do
    intPrev ← intensities.get(i − 1);
    retTPrev ← retentionTime.get(i − 1);
    mAct ← (intAct − intPrev)/(retTAct − retTPrev);
    if mAct <0.3*mPrev then
        return i;
    end
    intAct ← intPrev;
    retTAct ← retTPrev;
    mPrev ← mAct;
end
return i;
```

As mentioned before, the right limit is calculated using the same idea but moving along the right side. With that the peptide's start and the end limit are found.

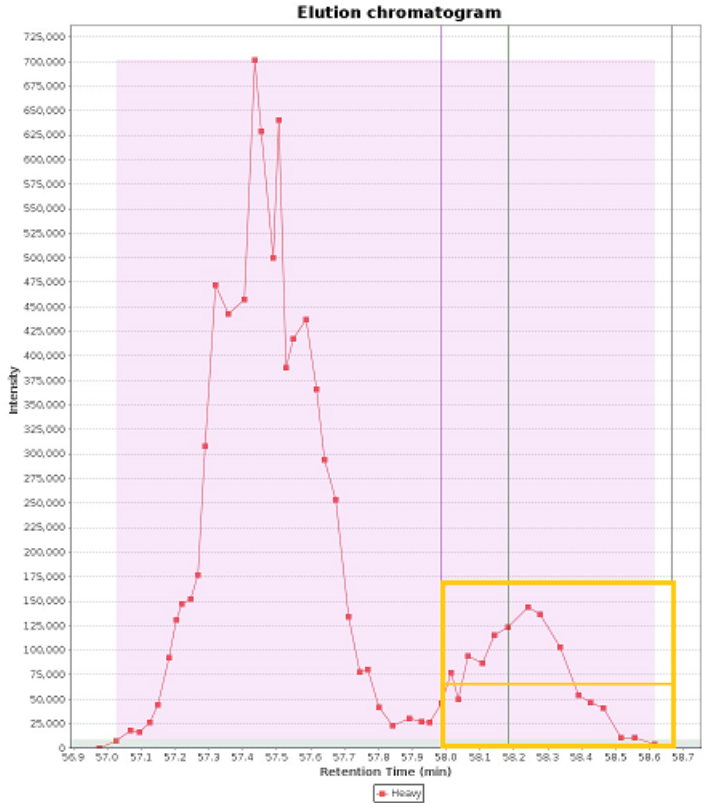


Figure 26: In this image is shown the original elution chromatogram with the limits found using the smoothing limits calculation. The resulting peak in time is squared in orange.

Only one peak shape:
If were applied a lot of smoothings but only one peak shape can be seen, the most probable thing is that there will be another peak mixed with the bigger one.

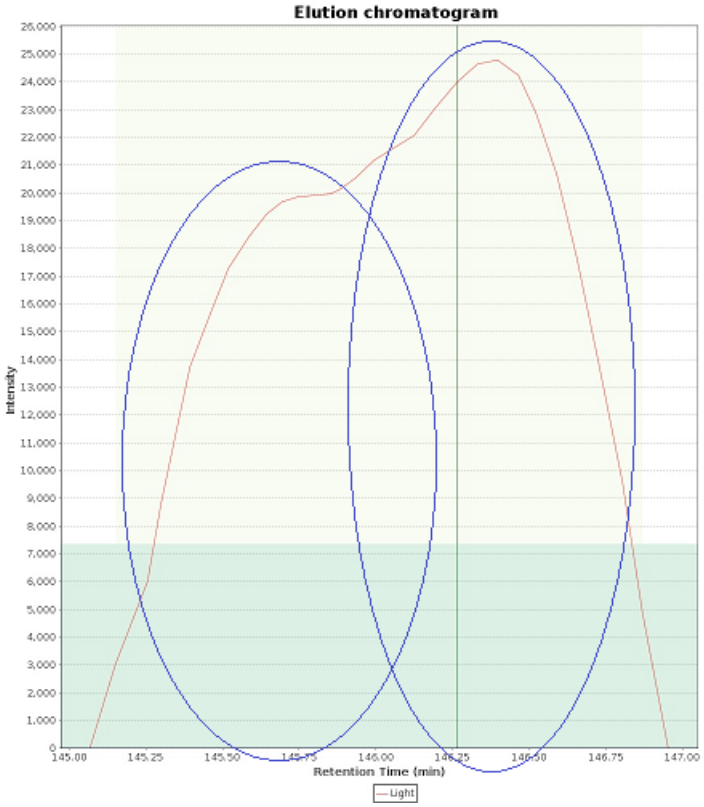


Figure 27: In this image is shown a smoothed peak in time where is detected only one shape but can be distinguished two (surrounded in blue circles).

For this reason, the algorithm will search for a clearly slope tendency change in order to detect this second peak.

First of all, as seen previously, the algorithm will search the nearest peak (top) position. Once it is found, the algorithm will check if the MS1 scan is at its left or right side.
If the MS1 scan is on its left side, the algorithm will search for the start point from the peak's top position to the right.

If while the algorithm is looking for the start point a tendency change before the MS1 scan is found, this will be the end point because another peak has been found mixed with the bigger peak. This step will be repeated until the MS1 is reached and after that the start point is found. This is because the

start point will always be at the MS1 scan left side and the end point at its right side.
The main idea is the same that when there are more than one peak, but in this case the algorithm is looking for the possibility of having more than on peak within a bigger one.

```
i ← startPos;
while start not found do
    tChange ← false;
    for i decreasing to 1 and not tChange do
        intPrev ← intensities.get(i − 1);
        retTPrev ← retentionTime.get(i − 1);
        mAct ← (intAct − intPrev)/(retTAct − retTPrev);
        if mAct <0.5*mPrev then
            if i >ms1pos then
                end ← i;
            else
                start ← i;
            tChange ← true;
        intAct ← intPrev;
        retTAct ← retTPrev;
        mPrev ← mAct;
    if start not found and i == 0 then
        start ← 0;
```

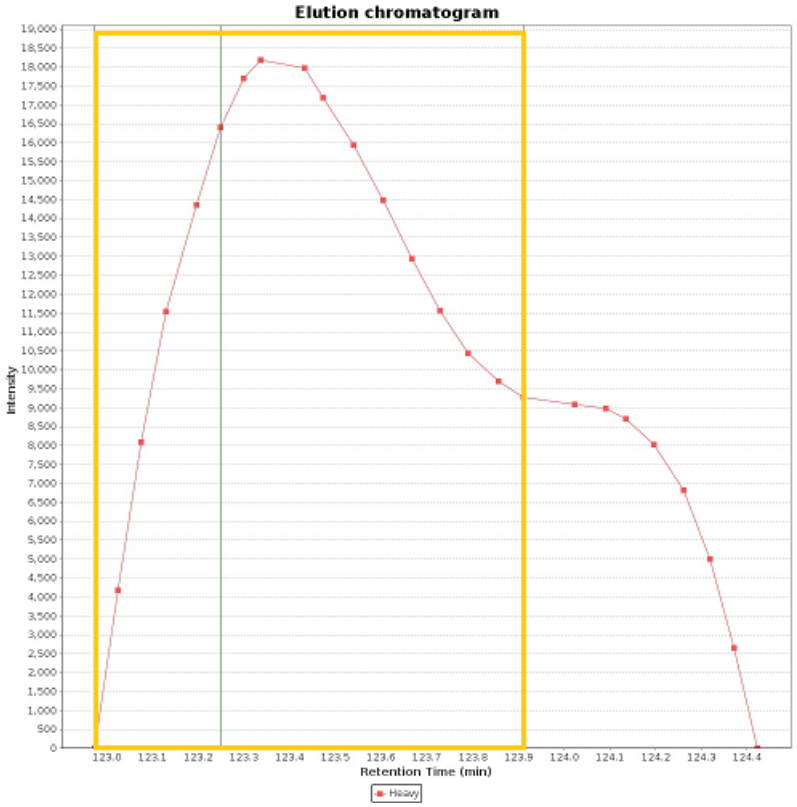


Figure 28: In this image, can be seen that the MS1 scan is at the right side. Following the explained algorithm, the algorithm will search for the start limit at the peak left side. As the first tendency change is after the MS1 scan, this point will be the start limit. For this reason, as no end limit has been found, the algorithm will look for it at the peak right side.

In the case where the MS1 scan is at the right side of the peak, the algorithm will look for the end limit. If the start limit has not been found while the algorithm was looking for the end, will look for it.
In order to search for the end limit, the algorithm will follow the same strategy than the one used with the start limit, but this time, moving along the right side and looking for the end instead of the start. Of course, as happened before, a start limit can be found while looking for the end if the MS1 scan has not been reached yet.

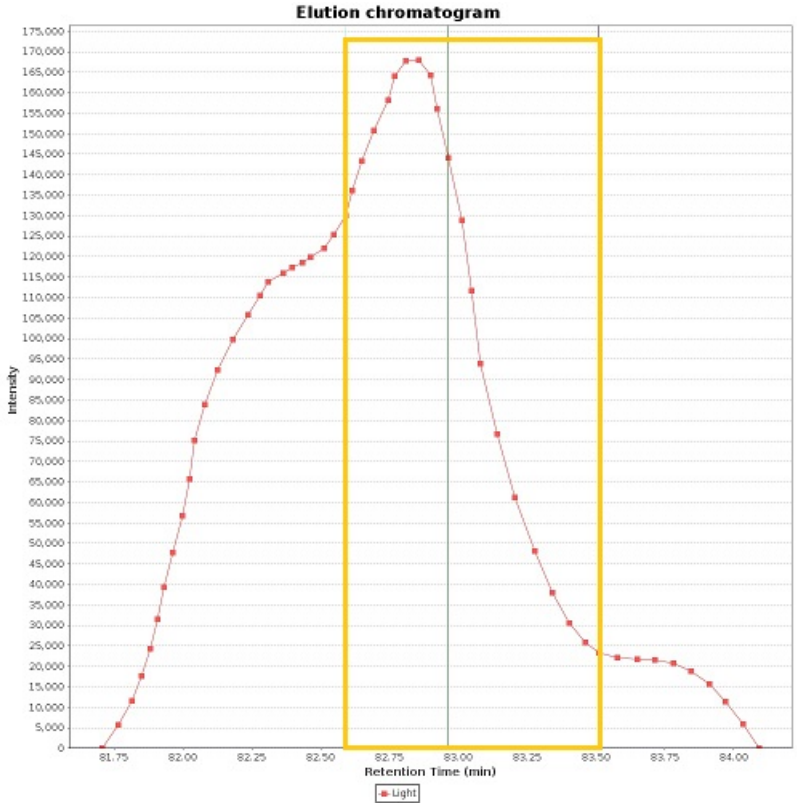


Figure 29: In this image, can be seen that the MS1 scan is at the right side of the peak. In this case, the algorithm will search the end limit at the peak's right side. Here the first tendency change is after the MS1, for this reason, the point will be the end limit and the start limit will be searched at the peak's left side.

In figure 30 is shown the result of the algorithm application. In these images can be seen the original elution chromatogram and the smoothed peak in time with the limits found (orange squares).

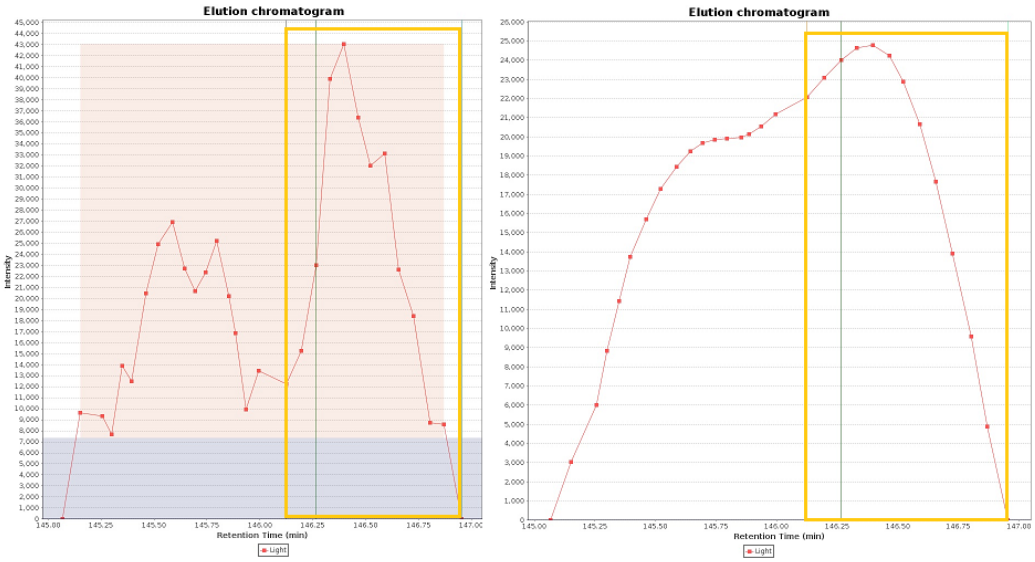


Figure 30: Original peak in time (left) and smoothed peak in time (right) showing the limits found using the smoothing limits algorithm.

PPM Score

Another peptide treatment is based on a 2008 article published in the Journal of Proteome Research[1] where is explained an algorithm that originally was thought to reject noise, but also performed well in deconvoluting signals from two different peptides with similar masses that eluted concurrently.

The authors stated that:

> "For each spectral peak within the mass window, the mass deviations gauging the precision of a particular peak with respect to its labeled (or unlabeled) counterpart were summed, both in that spectrum and adjacent spectra. This score was recursively determined for all spectral peak combinations in the mass window. Scores were normalized to generate a z-score in which higher values corresponded to peak sets with similar mass precision, suggesting that these peaks were likely of peptide origin."

Using this idea (adjacent peaks and label counterparts have a similar ppm variation), a score will be calculated for every retention time where the pep-

tide was found.

The algorithm is calculated as follows:
For every label of the peptide, the algorithm will get for every retention time their peak ppm variation, which is the difference between the original mz and the found mz (with the associated error). Once the algorithm has this ppm variation, will calculate the difference between that ppm variation and the ppm variation of the other labels. All these differences will be summed in order to produce part of the score, the other part will be calculated using the actual label adjacent peaks. This is, do the difference between the actual ppm variation with its previous and posterior ppm variations. Finally, these differences will be accumulated in the score obtained until now in order to obtain the average. Can be said that the score is the average of all those ppm variation differences.

Procedure calcPPMScore

Input: retentionTime: peptide retention time, labels: peptide labels
Result: ppmScore: PPM Score

```
scoreArr ← [];
defaultScore ← ppm/2;
for retT in retentionTime do
    score ← 0;
    counter ← 0;
    for label in labels do
        piT ← getPeakInTime(label);
        if exists retT in piT then
            actVar ← piT.getPPMVariation(retT);
            prevRetT ← getPreviousRetT(retT);
            if exists prevRetT in piT then
                prevVar ← piT.getPPMVariation(prevRetT);
                score ← score + |actVar − prevVar|;
            else
                score ← score + defaultScore;
            end
            postRetT ← getPosteriorRetT(retT);
            if exists postRetT in piT then
                postVar ← piT.getPPMVariation(postRetT);
                score ← score + |actVar − postVar|;
            else
                score ← score + defaultScore;
            end
            counter ← counter + 2;
            for lab in posterior labels do
                piT2 ← getPeakInTime(lab);
                if exists retT in piT2 then
                    actVar2 ← piT2.getPPMVariation(retT);
                    score ← score + |actVar − actVar2|;
                else
                    score ← score + defaultScore;
                end
                counter ← counter + 1;
            end
        end
    end
    scoreArr.push(score/counter);
end
ppmScore ← zScore(scoreArr);
```

Once the score for every peptide's retention time is obtained, the algorithm will calculate its z-score in order to have the score normalized. Will be calculated as follows:

Function zScore(score)

Input: score: PPM Score
Output: zScore: normalized PPM Score
$mean \leftarrow mean(score)$;
$stDev \leftarrow stDev(score)$;
$zScore \leftarrow []$;
for *every s in score* **do**
  $zScore.push((s - mean)/stDev)$;
end
return *zScore*;

The PPM Score lower (negative) values corresponds to peak sets with similar mass precision, while higher values (positive) corresponds to mass changes.

In figure 31 can be clearly seen how the PPM Score stays stable when the different labels and adjacent peaks have a similar ppm variation and how it increases when they change.

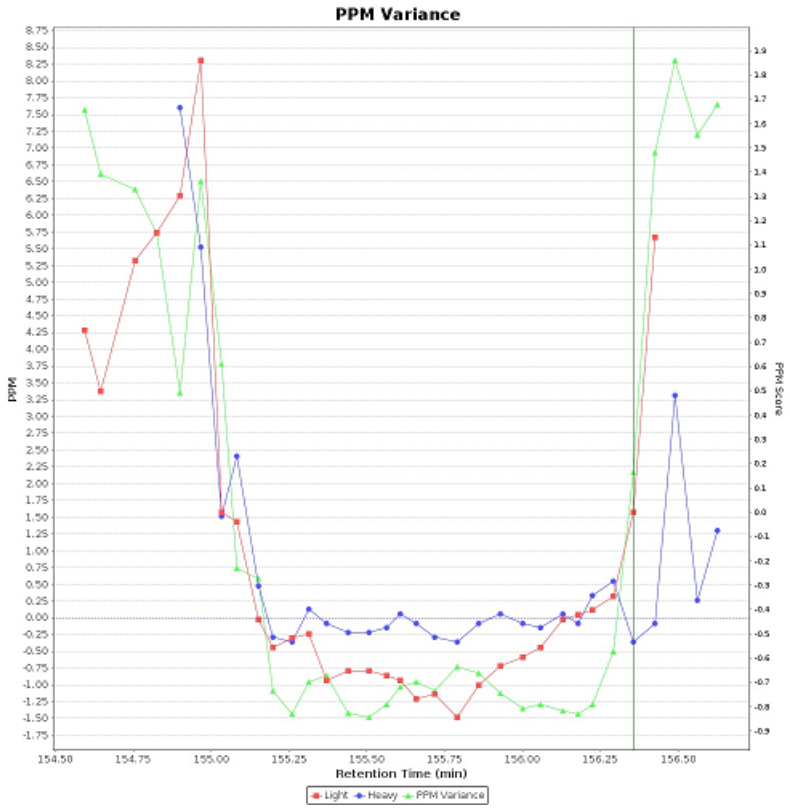


Figure 31: This image shows the peptide's ppm variation for the light (red) and heavy (blue) labels, and its corresponding PPM Score (green). The dark green line is the MS1 scan where the peptide was precursored.

Searching the peptide limits:
The PPM Score provides a way to calculate the Peptide boundings in order to calculate the area in a more precisely way.

The algorithm is divided in two different parts:

- Find the correct peak shape
- Find the peak shape limits

Finding the correct peak shape
In the previously explained image (figure 31), only one peak can be observed

(one negative score zone), but in some cases, can be clearly observed a couple of negative zones:

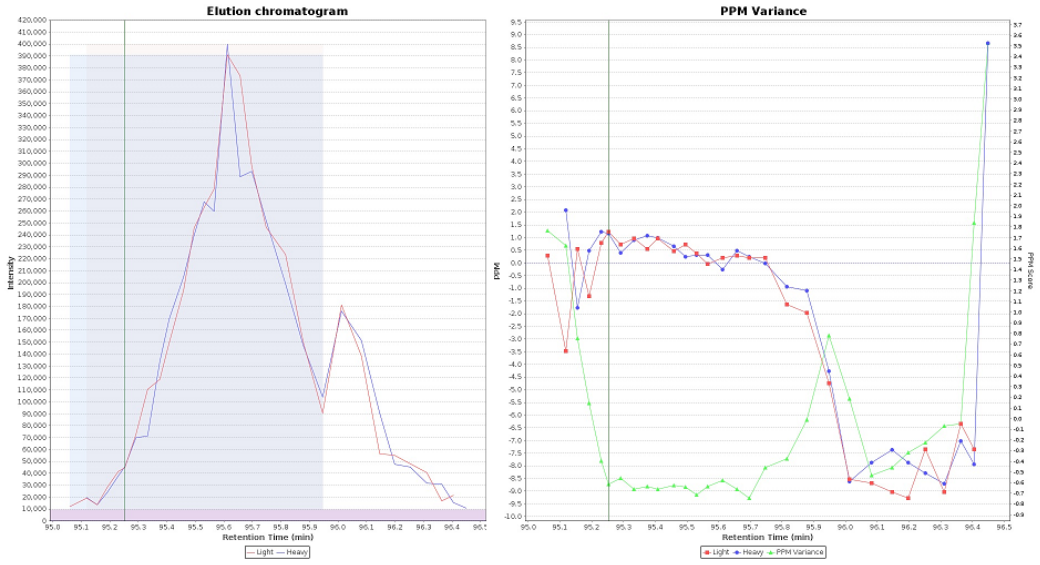


Figure 32: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image.

In figure 32 can be clearly seen on the left image two different peak shapes and on the right image a couple of negative score zones. Will always be considered that the correct peak will be the nearest to the MS1 scan retention time (vertical line), because as mention in previous chapters is the nearest scan to the precursor scan that had identified the peptide.

Finding the peak shape limits

In order to find the peptide's limits, four different algorithms were tried:

Algorithm 1: Negative scores

In this algorithm was decided to set the peptide's limits on the peaks where the score changes from negative to positive.

First of all the algorithm will have to find the negative zone where the ms1 scan retention time is, and move along its right and left side in order to find the start and the end limit peaks.

The problem of this idea was that sometimes this algorithm cut the real start and end of the peptide as can be seen below:

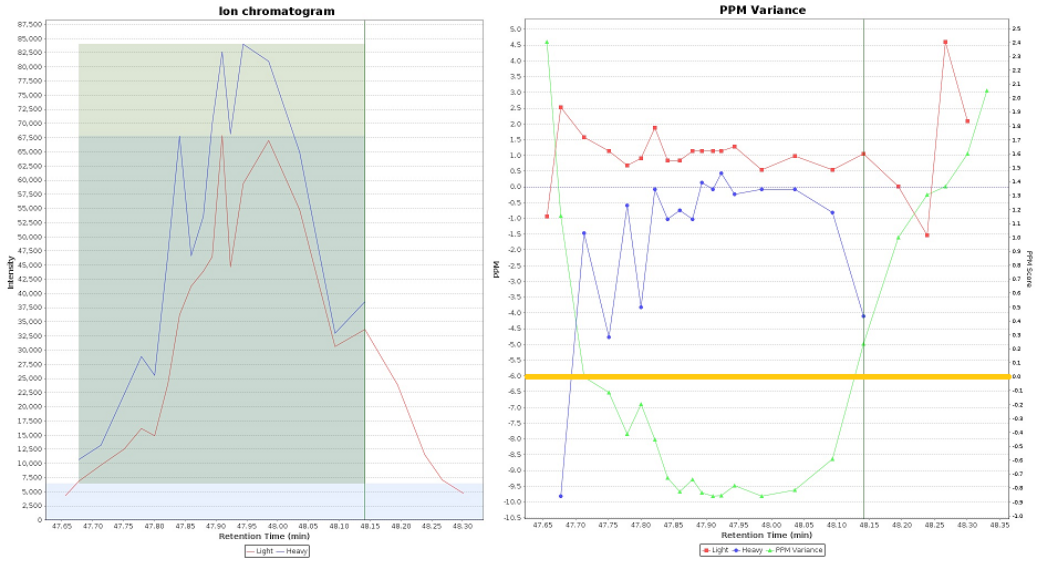


Figure 33: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image. The orange line indicates the ppm score limit, this is the point where the score goes from negative to positive. In these images can be seen that the peptide's bounds are smaller than what should be.

Algorithm 2: Priority on tendency

As in the first algorithm the boundings sometimes were established before time, a change in the PPM Score algorithm was introduced. Now, the difference between the actual peak and its adjacent peaks will have more priority than the difference between the other label counterparts.

As can be seen in figure 34, in most cases a better result than in the first algorithm was obtained:

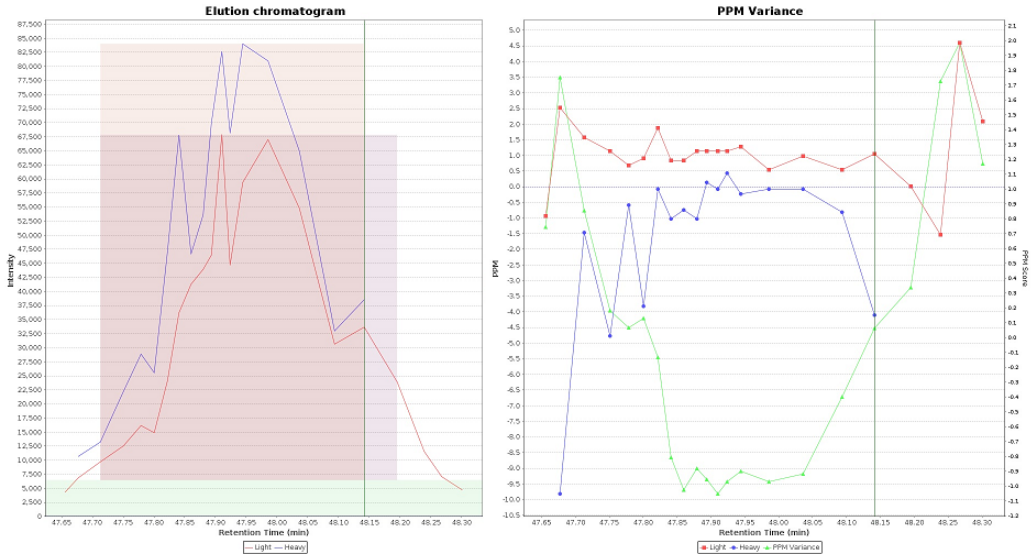


Figure 34: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image. The boundings are defined better than in Figure 33.

But in other cases, the algorithm didn't work as expected, still cut before time:

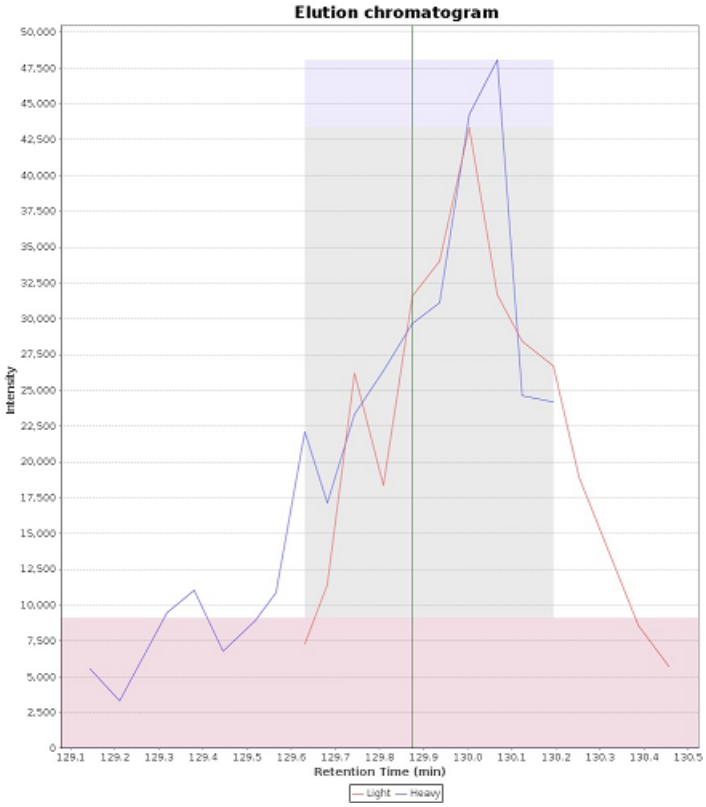


Figure 35: Peptide's Light (red) and Heavy (blue) peak in time where the limits of the peptide are squared and were obtained using the second algorithm. Can be seen that the limits where set before time.

Algorithm 3: Limit formula

In this case, was decided to establish a new limit different than zero in order to find the boundings, using a medium point between the minimum score and the maximum. This limit was calculated as follows:

$$limit = minScore + \frac{maxScore - minScore}{2}$$

This new version worked better than the previous ones, but was still cutting before time.

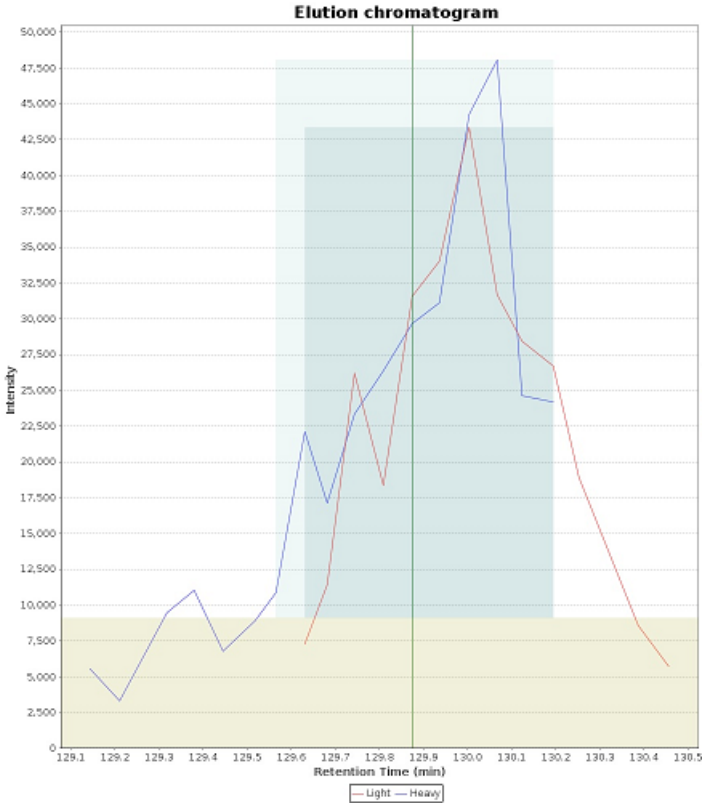


Figure 36: Peptide's Light (red) and Heavy (blue) peak in time where the limits of the peptide are squared and were obtained using the third algorithm. Can be seen that the limits where set before time.

Algorithm 4: Tendency change

After studying hundreds of different elution chromatograms with their corresponding PPM Score plots, was realized that the limits should be well established if the algorithm looks for a tendency change once it has moved until the positive numbers. This is, still looking for the start and end limits if the algorithm is on the negative score zone, or if it is on the positive score zone but it is still increasing (if is decreasing or is having the same value is a tendency change).

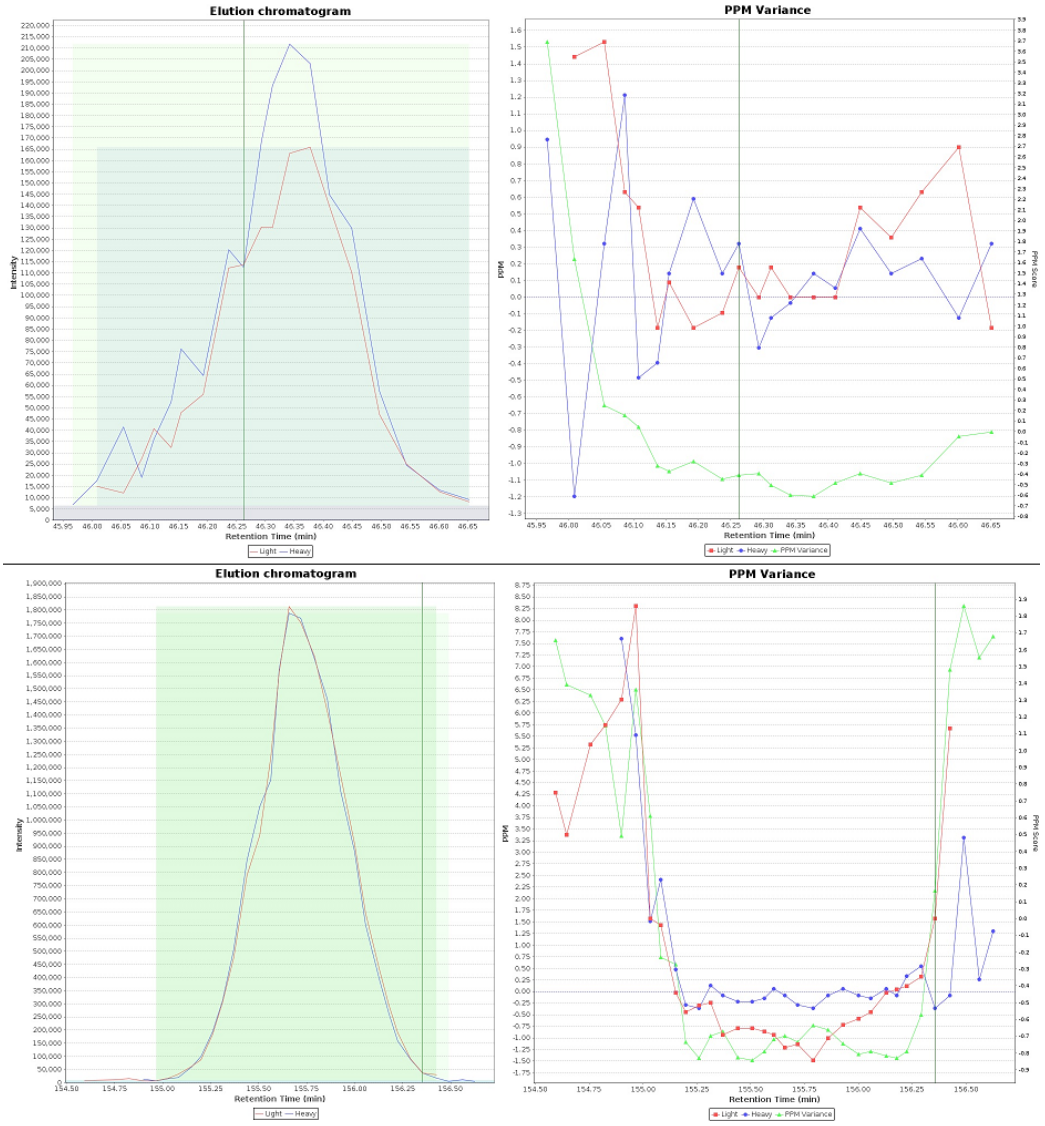


Figure 37: Peptide's Light (red) and Heavy (blue) peak in time on the left images, and their ppm variance and ppm score (green) on the right images. The limits are set on the tendency changes. On the top image there is not a tendency change so that the peptide is all the obtained peak in time, while in the bottom image there are two tendency changes: for the start limit and for the end limit.

In figure 37, a couple of examples had been added in order to show how the

tendency change is fitting well. Lets see that the algorithm works better than the previous ones using the previous example:

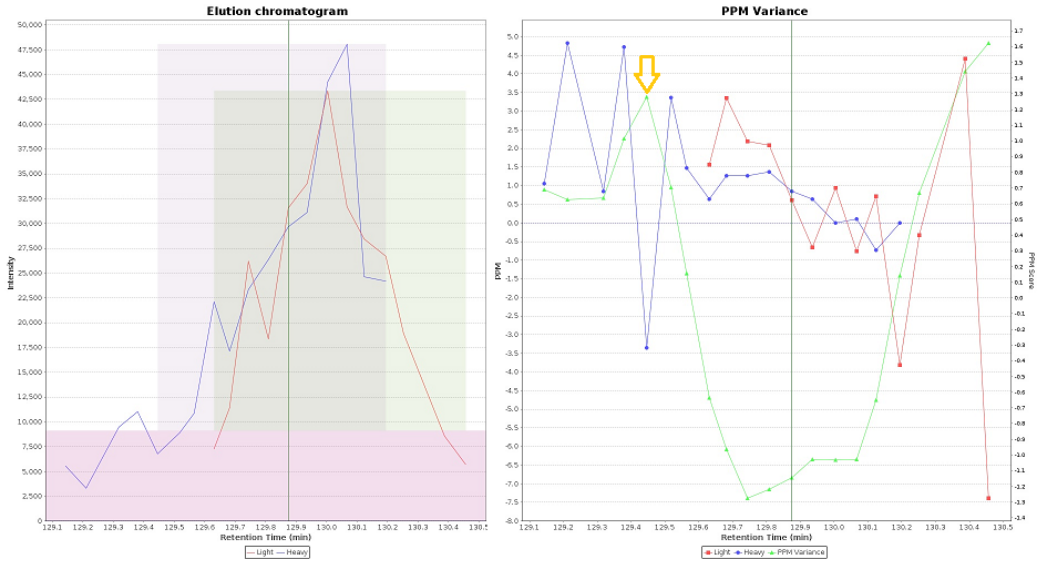


Figure 38: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image. The limits are well approximated using the tendency change algorithm (4th). An orange arrow indicates the point where can be found the start limit tendency change.

As the tendency change is the algorithm that has the best accuracy, this is the one that will be used in the PPM Score peptide treatment.

PPM Score limitations

Can be seen that the PPM Score is working well in these cases, but exists a couple of problems with this algorithm:

If in the PPM Score is found two different negative zones without a positive separation, the algorithm will think that there is only one peak.

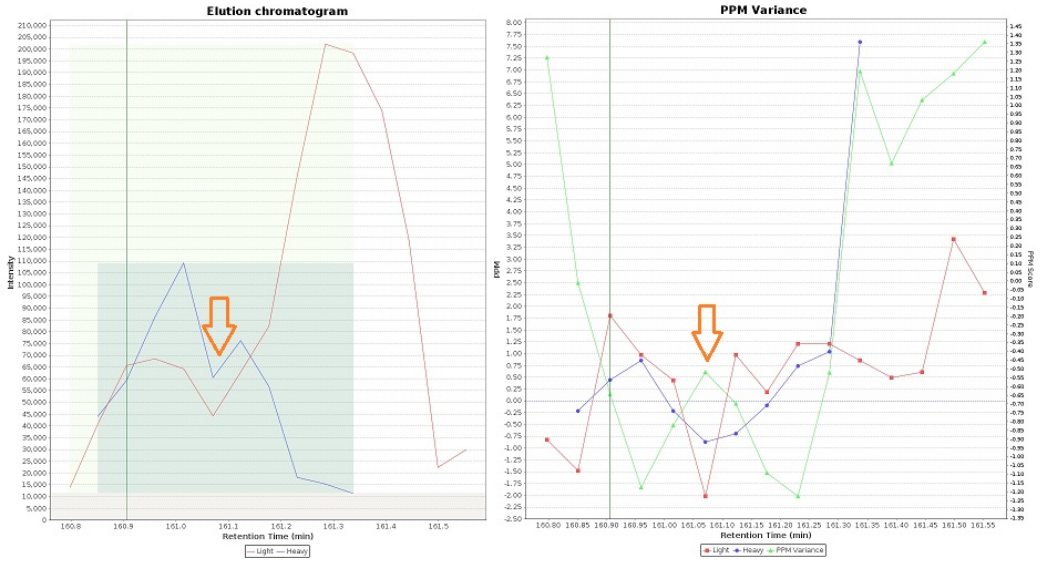


Figure 39: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image. The limits are not well approximated using the tendency change algorithm (4th). An orange arrow indicates the point where should be found the limit.

A possible future fix could be use the slope variance formed by the ppm score and the retention time (something similar to the technique used in the smoothing limits).

The other problem is that if the labeled counterpart has less retention time peaks, or is moved (starts before, starts after...), the PPM Score won't provide a good fitting. This is because the ppm variance between these labels is not similar ('is not the same peak').

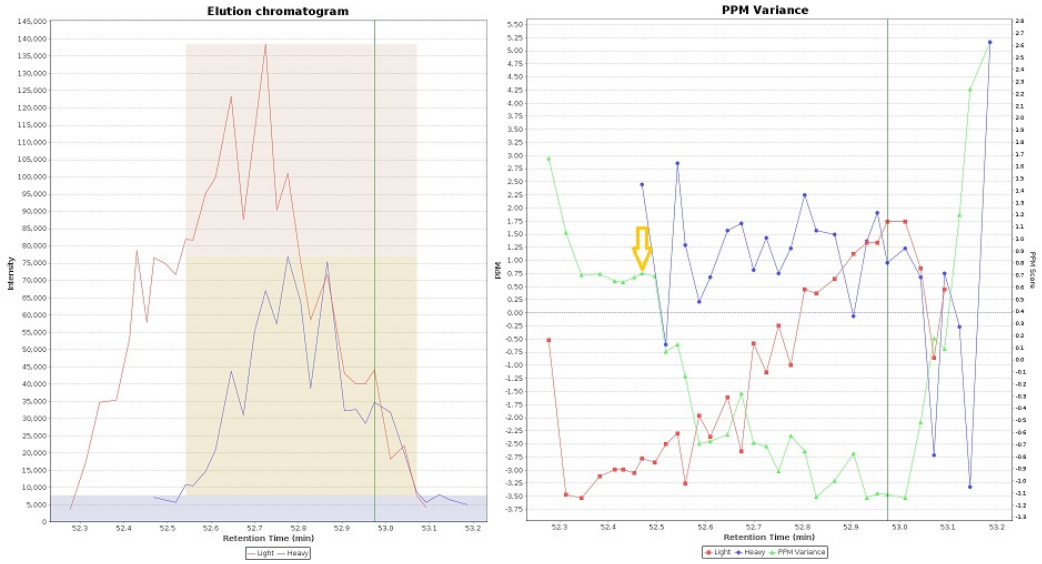


Figure 40: Peptide's Light (red) and Heavy (blue) peak in time on the left image, and their ppm variance and ppm score (green) on the right image. The limits are not well approximated using the tendency change algorithm (4th) because the dimension of the light peptide is different than the heavy one. The orange arrow indicates the point where the limit was established.

For this reason, in known cases where a label is moved respect its label counterparts, the PPM Score should not be used.

Reconstruction

The other peptide treatment is the reconstruction method, this method will be used in order to reconstruct peptides' peak in time where the start and end limit does not have zero intensities. This method can be combined with the smoothing limits or the ppm score in order to obtain a better shape.

The reconstruction will be done calculating the average retention time increment per peak found and the intensity increment per retention time step for the both sides of the 'bell' shape. This knowledge will be applied in order to obtain new fiction points. Lets see the algorithm idea:

Procedure reconstructPeakInTime

Input: retentionTime: peak in time retention time, intensities: peak in time intensities

Result: peak in time reconstructed

$retTIncL \leftarrow getRetTAvgIncr(left)$;
$intIncL \leftarrow getIntensityIncrPerRetTStep(left)$;
while *firstInt() >0* **do**
    $newRetT \leftarrow firstRetT() - retTIncL$;
    $newInt \leftarrow firstInt() - intIncL * retTIncL$;
    $putFirst(newRetT, newInt)$;
end
$retTIncR \leftarrow getRetTAvgIncr(right)$;
$intIncR \leftarrow getIntensityIncrPerRetTStep(right)$;
while *lastInt() >0* **do**
    $newRetT \leftarrow lastRetT() + retTIncR$;
    $newInt \leftarrow lastInt() - intIncR * retTIncR$;
    $putLast(newRetT, newInt)$;
end

The retention time average increment is obtained calculating from the top of the peak to its start or end, the existing retention time increment between every two points. In the other hand, the intensity increment per every retention time step is calculated doing the average of the division of the intensity increment and the retention time increment between every two consecutive points. This way the new intensity can be calculated using the last (or first) intensity and the retention time increment.

Using this algorithm in order to calculate the area, a better peak in time shapes are obtained.

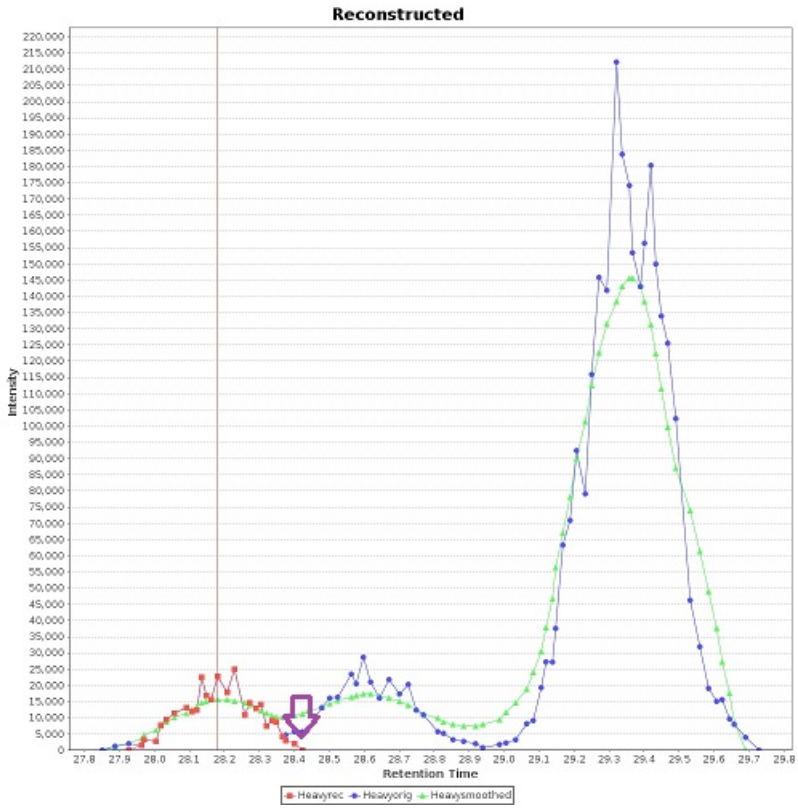


Figure 41: Peptide peak in time where can be observed the original peak in time (blue), the smoothed peak in time used in order to find the limits (green) and the reconstructed peak in time (red). The purple arrow indicates the right side reconstruction that has been done.

3.1.6 Area

Once the peptide has been found and has been applied all the peptide treatments (if it was considered to), the area formed by the peak in time of every label must be calculated in order to provide the needed ratios.

This area can be calculated using two different methods:

- Approximating the peak in time to trapezoids.
- Approximating the peak in time to a gaussian function.

Trapezoid method
This method consists in calculate the peak in time area drawing rectangles between two consecutive points and summing the resulting area. This rectangle is drawn using the retention time as the rectangle base and the intensity as the height. In order to fill the existing gaps between the rectangle and the peak in time shape a triangle will be drawn, and the resulting area will be summed as well.

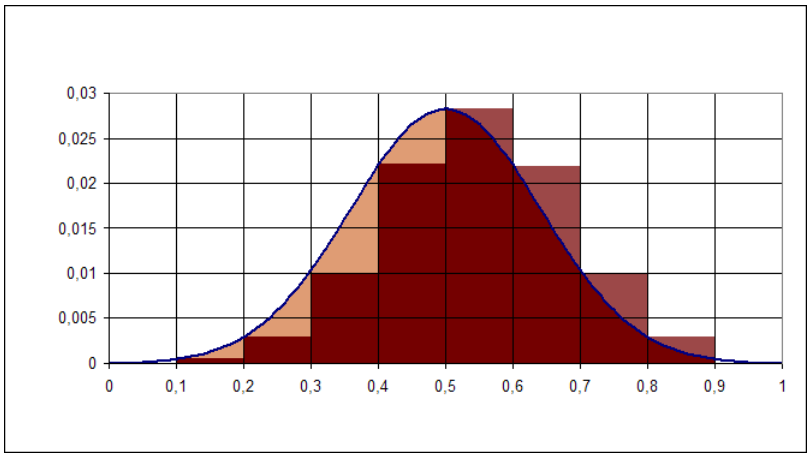


Figure 42: Using rectangles and triangles in order to calculate a bell shape area.

As can be seen in the previous figure, a rectangle has been formed every two points from which can be calculated the area.

$$\begin{aligned} area_{rectangle} &= base * height \\ &= (retT_2 - retT_1) * intensity_1 \end{aligned}$$

At this point a triangle will be drawn in order to fill the existing gap (shape's left side) or subtract it from the previously drawn rectangle (shape's right side). In order to do it, the next formula will be used:

$$\begin{aligned} area_{triangle} &= \frac{base * height}{2} \\ &= \frac{(retT_2 - retT_1) * (intensity_2 - intensity_1)}{2} \end{aligned}$$

Note that this formula can be used in both shape's sides. At the shape's left side, the resulting area will be positive so that the area will be summed, while at the shape's right side, the resulting area will be negative so that the area will be subtracted from the previously calculated rectangle.

This method will provide a good approximation to the bell shape.

Gaussian function

The other method is to approximate the existing bell shape to a gaussian function in order to calculate the area doing its integral.

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

Where a is the peak height, b the center of the peak, and c is the variable that controls the bell's width.

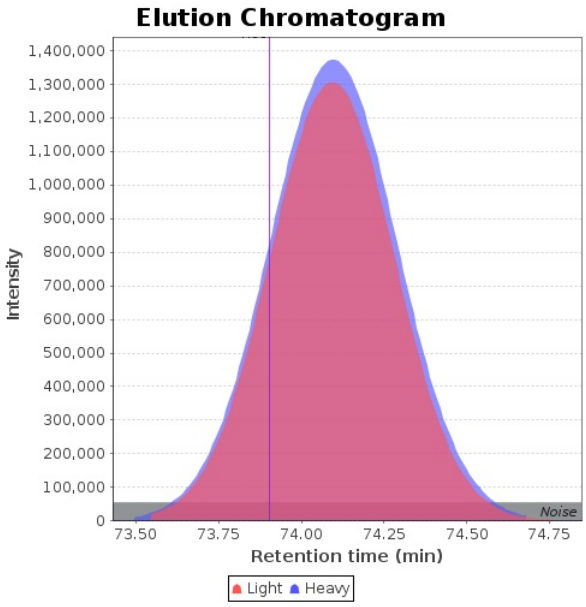


Figure 43: Light and heavy peptides approximated to a gaussian function.

Calculating a

a is the easiest variable to be calculated, this is the peak in time maximum intensity. In order to determine all peaks (points) will be checked.

Calculating b

One could think that the center of the peak is the retention time where the maximum intensity was found, this is a wrong supposition because the given peak in time is not a perfect bell shape. In order to calculate the bell center, the full width at the half maximum (FWHM) will be calculated because its center point will be b.

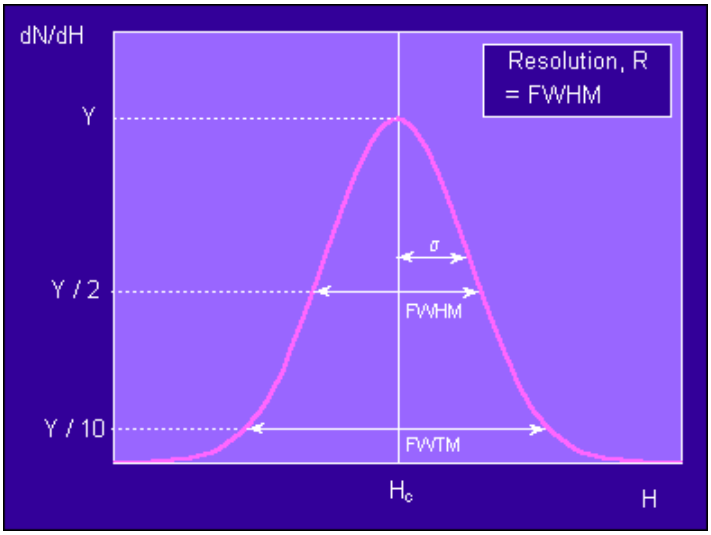


Figure 44: Gaussian bell parts.

In order to calculate the needed FWHM will be necessary to find the bell's left and right point where the intensity is the half of the maximum intensity. The existing problem is that maybe these points do not exist in the given peak in time, if it is the case, the points will be approximated. In order to do it, the first step is to obtain the two points where the half intensity should be between, once they have been obtained the corresponding retention time can be calculated. The next steps will be:

1. Calculate the intensity increment between these two points:

$$\triangle_{intensity} = |intensity_1 - intensity_2|$$

2. Calculate the retention time increment between these two points:

$$\triangle_{retT} = |retT_1 - retT_2|$$

3. Calculate the intensity increment per retention time increment:

$$\triangle_{intensityTime} = \frac{\triangle_{intensity}}{\triangle_{retT}}$$

4. Calculate the left retention time increment that should be added to obtain the half-maximum intensity retention time:

$$retT_{toAdd} = \frac{intensity_{toSearch} - intensity_1}{\triangle_{intensityTime}}$$

5. Obtain the half-maximum intensity retention time:

$$retT = retT_1 + retT_{toAdd}$$

6. Repeat 4 and 5 for the shape's right side.

7. Calculate the FWHM:

$$FWHM = retT_{left} - retT_{right}$$

8. Calculate the center of the bell:

$$b = retT_{left} + \frac{FWHM}{2}$$

Calculating c

c can be calculated using different concepts: FWHM relation, FWTM (Full width tenth to minimum) relation... As the FWHM has been previously calculated, will be the method used in order to find c.

The existing relation between FWHM and c is:

$$FWHM = 2\sqrt{2ln2}c$$

so that:

$$c = \frac{FWHM}{2\sqrt{2ln2}}$$

Now that all the gaussian function variables have been obtained, the area can be calculated doing the function integral:

$$area = \int_{-\infty}^{\infty} ae^{-\frac{(x-b)^2}{2c^2}} \, \mathrm{d}x = ac\sqrt{2\pi}$$

Applying the previous formula the peak in time area will be calculated.

3.1.7 Output

After the area calculation, the algorithm stores the quantification output data. This data will be:

Search id Peptide's search algorithm number identifier.

Run id Peptide's mass spectrometer run identifier.

Peptide id Peptide's identifier.

MS2 scan MS2 scan where the peptide was precursored.

Noise Peptide's noise calculated.

Finding score Score that determines if all labels were well found.

For every label this information is provided as output:

sn Signal to noise calculated as follows:

$$sn = intensity_{max}/noise$$

First scan Peak in time first scan number.

Last scan Peak in time last scan number.

Number of scans Peak in time total number of scans.

Score Peak in time quality score.

Area Peak in time area.

Maximum intensity Peak in time maximum intensity.

Maximum intensity scan Scan number where the maximum intensity of the peak in time was found.

Maximum intensity retention time Retention time where the maximum intensity of the peak in time was found.

Peaks Peak in time peaks encoded in base64.

For every pair of the Light label and another one:

Area ratio Area ratio calculated as follows:

$$ratio_{area} = log_2(label_{area}/Light_{area})$$

Minimum area ratio Area ratio calculated using the other label area without the proportional noise area and the Light area. An area without noise is calculated as follows:

Trapezoid method: calculating the rectangle formed by the intensity equals to the noise and the corresponding retention time where they should be, and subtracting it from the total area.

Gaussian method: calculating the retention time where the intensities should be and calculating this area using the trapezoid method. Finally, this noise area will be subtracted from the gaussian area. The formula to find the retention time is:

$$retentiontime = \frac{2b \pm \sqrt{4b^2 - 4 * (b^2 + ln(intensity/a) * 2c^2)}}{2}$$

Maximum area ratio Area ratio calculated using the other label area and the Light area without the proportional noise area.

Intensity ratio Ratio between their maximum intensities.

Minimum intensity ratio Is the relation between the other label intensity without noise and the Light intensity.

Maximum intensity ratio Is the relation between the other label intensity and the Light intensity without noise.

3.2 Quantification methods

The proteomics quantification algorithm has three different quantification actions: batch quantification, cross quantification and single quantification.

3.2.1 Batch quantification

This method is the most common action for the quantification algorithm. The batch quantification quantifies a list of peptides.

The minimum needed information for every peptide is: the peptide sequence, the MS2 scan where was found, and the peptide charge.

The quantification algorithm works as follows: the algorithm starts loading the Mass spectrometer's output file into the algorithm's structures, then the peptides list is loaded and for every contained peptide the quantification is done following the previously explained quantification steps (peptide finding, peptide quantification, output). Finally, once all peptides have been quantified, the quantification output is written.

3.2.2 Cross quantification

The cross quantification is a technique where different runs for the same sample are done. This is done in order to obtain more MS2 scans in some samples and more MS1 scans in the others. As the MS2 are the scans where the peptides are detected, as much MS2 scans are found more peptides will be detected. But the peak in time information is found in the MS1 scans, so

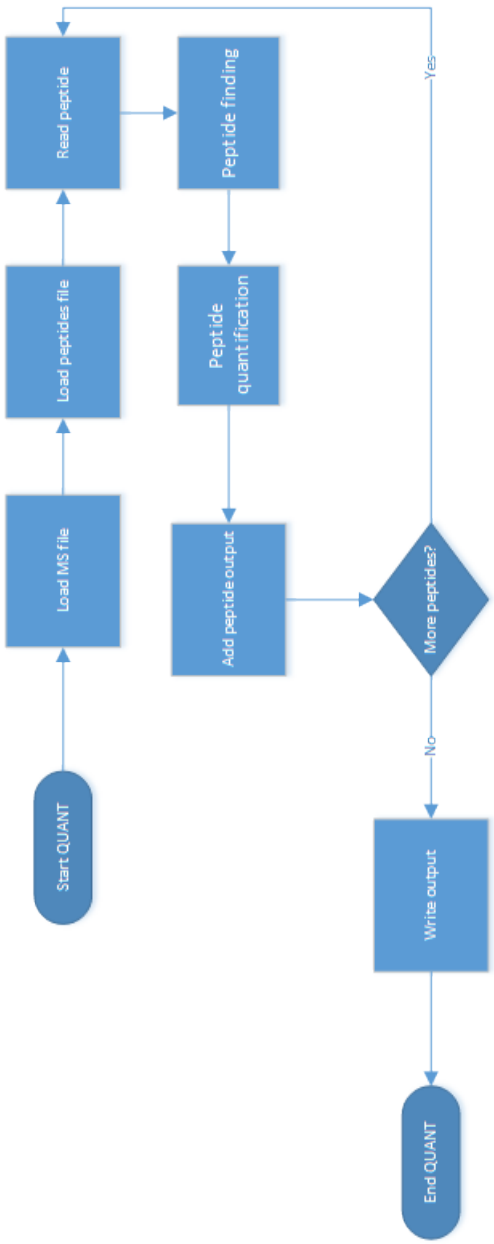


Figure 45: Batch Quantification algorithm blocks diagram.

the detected peptides (source) will be searched in the MS1 runs (target).

An example could be to perform a run using only the Light label so that more peptides will be detected, then perform another run with all the labels included. This way the algorithm will get the peptides from the first run (as only the Light label was used, more peptides could be detected) and will search all these peptides for every label in the second run.

The minimum needed information for every peptide in order to perform the cross quantification is: the peptide sequence, the MS2 scan where was found, and the peptide charge.

In this algorithm, every target run will be seen as a different cross quantification execution, so that all source peptides will be searched within it; For every target run, the algorithm quantifies its own detected peptides and after that starts with the source peptides quantification. As the algorithm does not know the MS2 scan where the source peptide can be found in the target run, the algorithm searches the source and target common peptides in order to create a regression that will estimate in which scan of the target file the peptide should be found. This regression will tell where to search every non-common peptide.

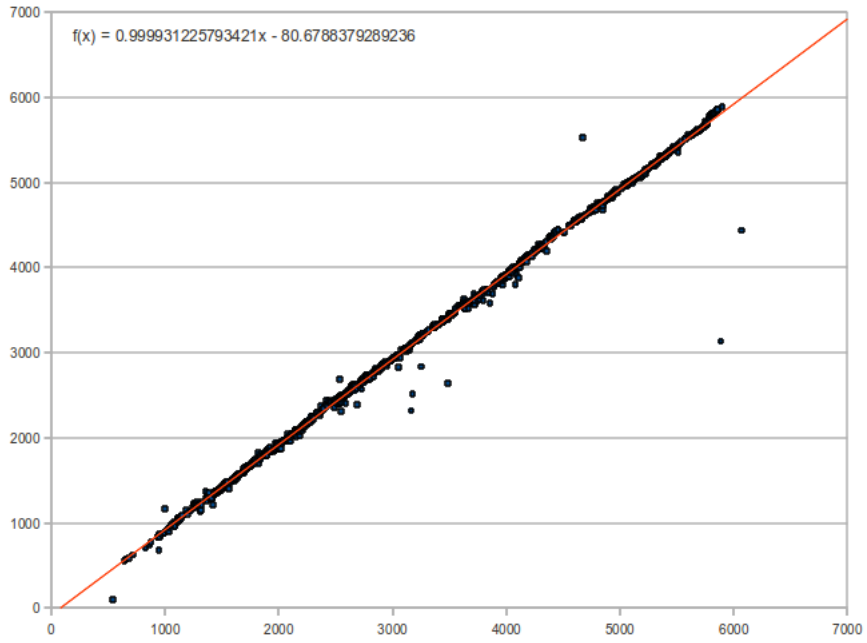


Figure 46: Cross quantification regression, that permits to locate the source peptides in the target run.

In order to calculate the regression, the common peptides are randomly divided in two different sets: a training set, and a test set. The training set will be used in order to create the regression while the test set will be used to validate it. To validate the obtained regression, the source peptide precursor scan in the target file will be calculated using the regression, then this scan will be compared with the known target precursor scan (the peptide was common in both runs). The existing difference error between the two obtained scans can be used in order to adjust a permissive window to search the peptides within it. This window is obtained calculating the median of these scan differences.

Once the regression is finally calculated, the algorithm estimates the precursor scan for every non-common peptide given and if it is found inside a concrete window, the algorithm will quantify it.

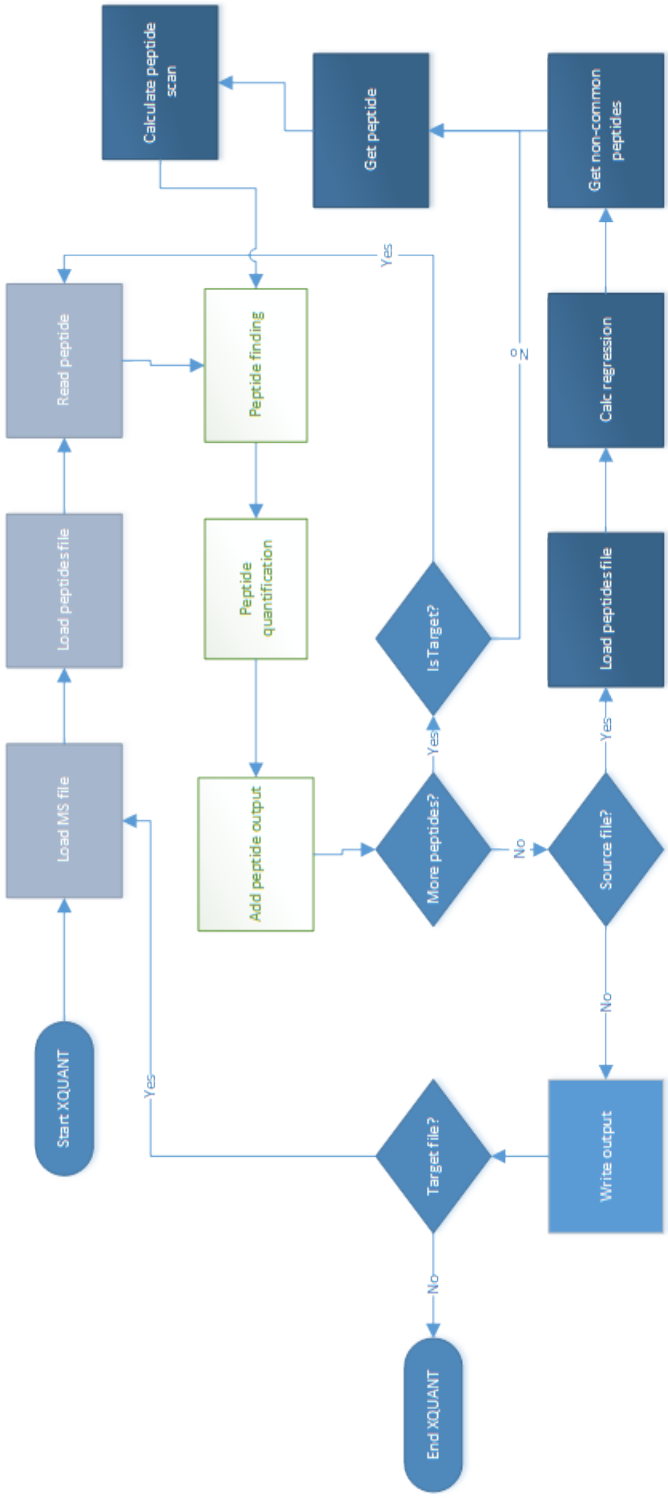


Figure 47: Cross Quantification algorithm blocks diagram.

3.2.3 Single quantification

The single quantification can be done in two different ways: setting only one peptide in the batch quantification list, or use the known information of the peptide obtained in a previous quantification of it. These known information are the peptide's peaks encoded in base64 that the quantification algorithm outputs.

The minimum peptide information needed is: the peptide sequence, the MS2 scan where was found, the peptide charge, the peptide noise, and for each label the encoded peaks.

This algorithm will directly decode and quantify, for each label, the peak in time obtained from the encoded peaks. There is no need of peptide finding process, the peaks are already known, so that the Mass spectrometer's output file will not be loaded.

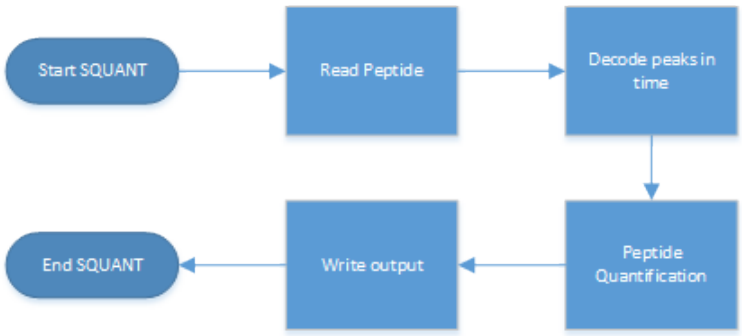


Figure 48: Single quantification algorithm blocks diagram.

3.3 Algorithm costs

In order to calculate the algorithm cost, it have to be splitted in different sub-algorithm and calculate their costs.

The first action is the peptide searching, this can be done in two different ways: setting a scan window to search within, or search until the peptide is not found in x consecutive scans. In the first option the worst case is perform the complete window check, so that the algorithm cost is $O(w)$ where w is

the scan's window size. In the second option, the worst case (completely improbable) is search the peptide within the whole file, so that the algorithm cost is $O(s)$, where s is the number of scans contained in the file.
Every scan has a number of ions within, lets consider i. The worst case in the peptide search within the scan is have to do the complete traversal. These peptide search is done for every isotope of the indicated isotopic distribution, lets consider d, so that the complete peptide searching algorithm will have a cost of $O(w \cdot i \cdot d)$ or $O(s \cdot i \cdot d)$.

At this point can be done two different things: perform a peptide treatment or not. If the treatment is not done, the additional cost will be 0. But if it is done will depend on the treatment.

- Smoothing: this kind of peptide treatment does a complete traversal of the peaks (same number than the scans) the number of times indicated by the user (t), so that its cost will be $O(s \cdot t)$ or $O(w \cdot t)$. If the limits are searched the cost will be $O(s \cdot (t+1))$ or $O(w \cdot (t+1))$.
- PPM Score: in this case the traversal is done three times: for the PPM Score calculation, the z-score calculation, and the limits finding. This way the algorithm cost will be $O(3 \cdot s)$ or $O(3 \cdot w)$.
- Reconstruction: the worst case will be do the complete peak in time traversal, so that the algorithm cost will be $O(s)$ or $O(w)$.

The area calculation is done with a complete peak in time traversal for the trapezoid method. The gaussian function will be calculated in the worst case doing the complete traversal too. So in both cases the cost will be $O(s)$ or $O(w)$.

All these processes are done for every given label (l) and peptide so that this process will be repeated $p \cdot l$ times.

In conclusion the most simple quantification will have a cost of $O(p \cdot l \cdot (w \cdot i \cdot d + w))$ or $O(p \cdot l \cdot (s \cdot i \cdot d + s))$.
If a peptide treatment is done can be obtained the next costs:

- Smooth: $O(p \cdot l \cdot (w \cdot i \cdot d + w \cdot t + w))$ or $O(p \cdot l \cdot (s \cdot i \cdot d + s \cdot t + s))$.

- Smooth and Limits: $O(p \cdot l \cdot (w \cdot i \cdot d + w \cdot (t+1) + w))$ or $O(p \cdot l \cdot (s \cdot i \cdot d + s \cdot (t+1) + s))$.
- Reconstruction: $O(p \cdot l \cdot (w \cdot i \cdot d + w + w))$ or $O(p \cdot l \cdot (s \cdot i \cdot d + s + s))$.
- PPM Score: $O(p \cdot (l \cdot (w \cdot i \cdot d + w) + 3 \cdot w))$ or $O(p \cdot (l \cdot (s \cdot i \cdot d + s) + 3 \cdot s))$ because the PPM Score is calculated only once for every peptide.

4 Algorithm design

In this section will be explained the interactions that the user should be able to perform in the algorithm and the architecture used in order to accomplish it. Finally, a class diagram will be shown in order to be able to explain the class interactions.

4.1 Use cases

The proteomic quantification algorithm must provide three different quantification actions: Batch quantification, Cross quantification, and Single quantification.

The batch quantification is the complete proteomics quantification, this means that the algorithm will quantify a set of peptides (usually the search algorithm output). The cross quantification algorithm searches the peptides that have been precursored in a Mass spectrometer run into another run. Finally, the single quantification is a unique peptide quantification, and can only be used when the batch quantification is done. The user could ask for the peptide chromatogram.

The user should be able to provide some configuration and preference parameters in order to adjust the algorithm.

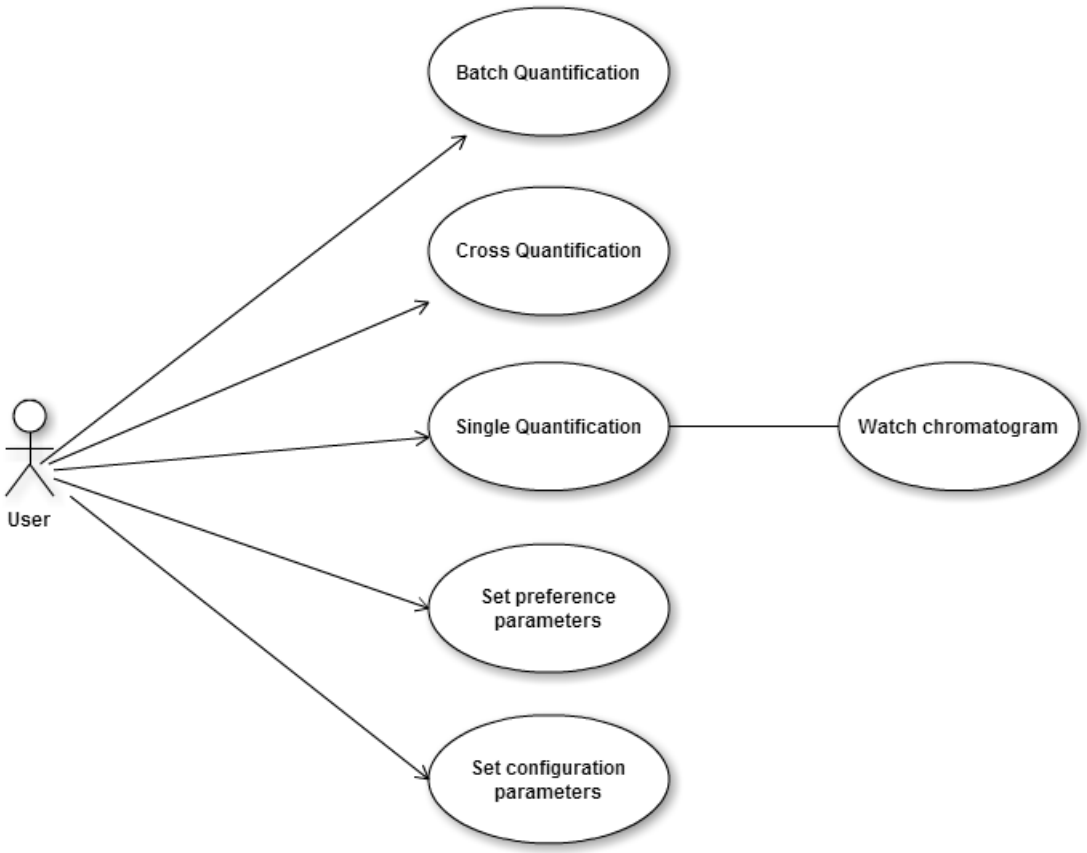


Figure 49: Use case diagram.

This configuration and preference parameters will be provided using a configuration file. In this file will be indicated the labels, the quantification algorithm to be performed and the related files too.

4.2 Software architecture

The software architecture is the system's components structure, and the relation between them.

In order to implement the proteomic quantification algorithm, a 2-tier architecture was used: Logic and Data tiers.

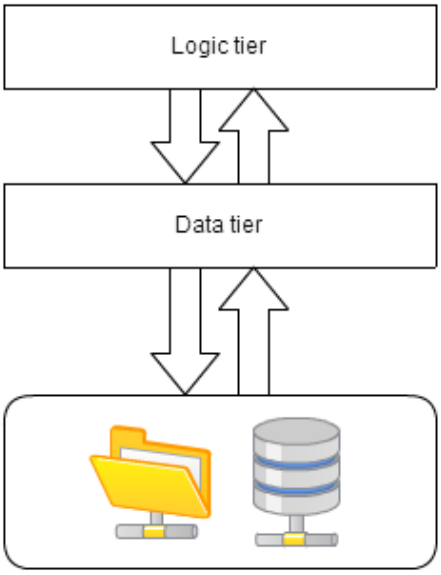


Figure 50: 2-tier architecture.

This architecture decision was made in order to obtain the maximum benefit of the properties desired for the algorithm. These properties are:

- Extensibility: Is one of the most important properties; the algorithm must be easily extensible in order to implement new features in a near future.
- Maintainability: The algorithm should be easily maintained by other programmers. The algorithm must be well designed in order to detect problems origins.
- Portable: The algorithm must be portable to other systems and platforms.
- Reusability: for other projects could be interessant use some parts of the algorithm.
- Reliability: a good design ensures a sturdiness grade that will prevent problems in the future.

This 2-tier pattern is used in order to perform a 3-tier pattern in conjunction with other platforms. The algorithm has no presentation tier, it is only an

algorithm, but in conjunction with other platforms should be seen as an extension or module, and for this reason, will require a presentation tier. This way, the platform that will have the quantification algorithm as a module will have the visualization tier, and this one will be interacting with the algorithm's domain tier.

This architecture has as advantage that completely separates every layer from the others. Doing so, the algorithm will be easily extensible (only a few components will be added to layers in order to add a new functionality). This architecture has the maintainability advantage too. The existing separation between layers prevents from errors: when an error occurs, only one layer should be modified in order to fix it.

Logic tier
The logic or domain tier is the layer that executes the actions demanded by users. This layer contains the algorithm itself with its three variants: batch quantification, cross quantification and single quantification. This layer manages the data and does the needed computations, but will never have into account the data storage or its presentation.

Data tier
This layer is in charge of the data retrieving and storage. This layer does the maintainability and integrity of the data too. In this quantification algorithm, the data layer provides an easily extensible way to add new input and output data formats. The algorithm (located in the logic tier) will be completely independent of the input or output data.

4.3 Class diagram

In this section the proteomics quantification algorithm class diagram will be explained. As mentioned before, the algorithm uses a 2-tier architecture. This architecture can be distinguished in the class diagram, where two different packages represent the tiers: Domain and Data packages. Every layer (package) has its own controllers in order to provide communication between tiers.

Data Package
The data package (tier) has five different functionalities:

- **Load mass spectrometer file**
  The mass spectrometer file has a concrete format, usually XML format, that should be parsed in order to retrieve the contained data. This algorithm initially uses mass spectrometer mzXML files, but is completely prepared to support different formats.
  After parsed the MS data, a set of Scans (level 1 and 2) is obtained, compound among other things by Peaks. With this data, the algorithm can perform the peptide searching. To do so, the data package will have the MsDataController.

- **Load the peptides that will be quantified**
  The peptides data can be loaded from different format files. This algorithm initially uses CSV files, where every row will contain a peptide. The algorithm is completely prepared to add and support different formats.
  The most important peptide data is: the peptide sequence, charge and precursor scan. This information can be obtained using the PeptideInputController.

- **Read the user's configuration file**
  The configuration file will be loaded using the ConfigurationController.

- **Write the quantification output**
  Initially the algorithm will store the quantification output data in CSV format, where every peptide quantification will be stored in a different row. To do so, the Data packages has the OutputController.

- **Draw the elution chromatogram**
  One of the algorithm functionalities is provide the elution chromatogram in single quantifications. To do so, the Data package has the Chart2D controller.

Logic Package
The logic class diagram contains the structure explained in the previous concepts chapter. In a mass spectrometer run, the peptide can be found with

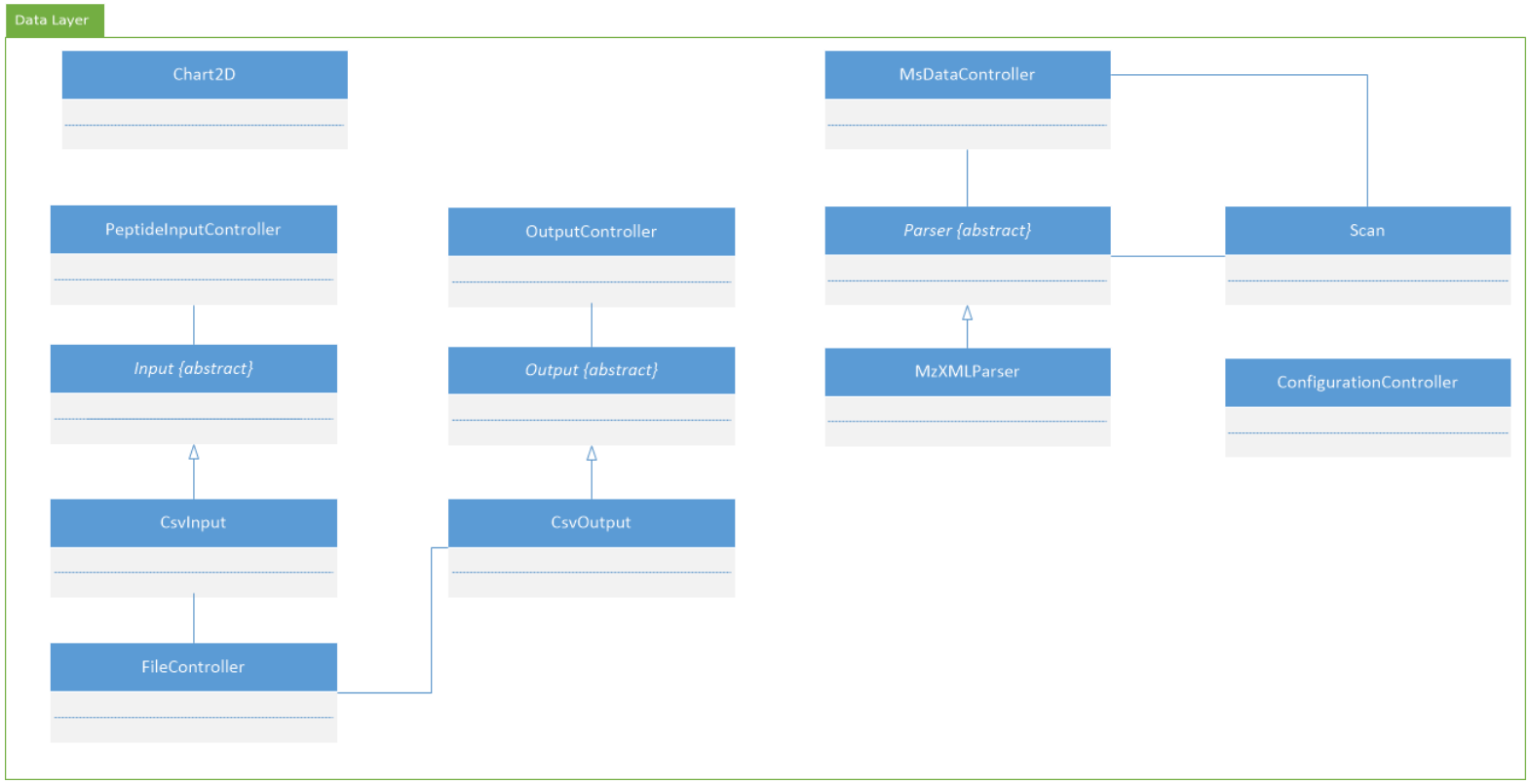


Figure 51: Proteomics quantification algorithm Data Layer.

different labels, and every label is composed by a set of Spectras, which at the same time, are composed by an isotope distribution of Peaks. Using the same isotope peak in every spectra, the labeled peptide shape can be retrieved sorting the peaks by retention time. This is the PeakInTime. Every different sub-algorithm used in order to do a peptide treatment for its quantification, will be represented as a different kind of PeakInTime.

Other packages
In the proteomics quantification algorithm there are two more packages: Auxiliar which have util functions for the algorithm and the needed classes to draw charts, and the Configuration package which contains the configuration and preferences set by the user's configuration file.

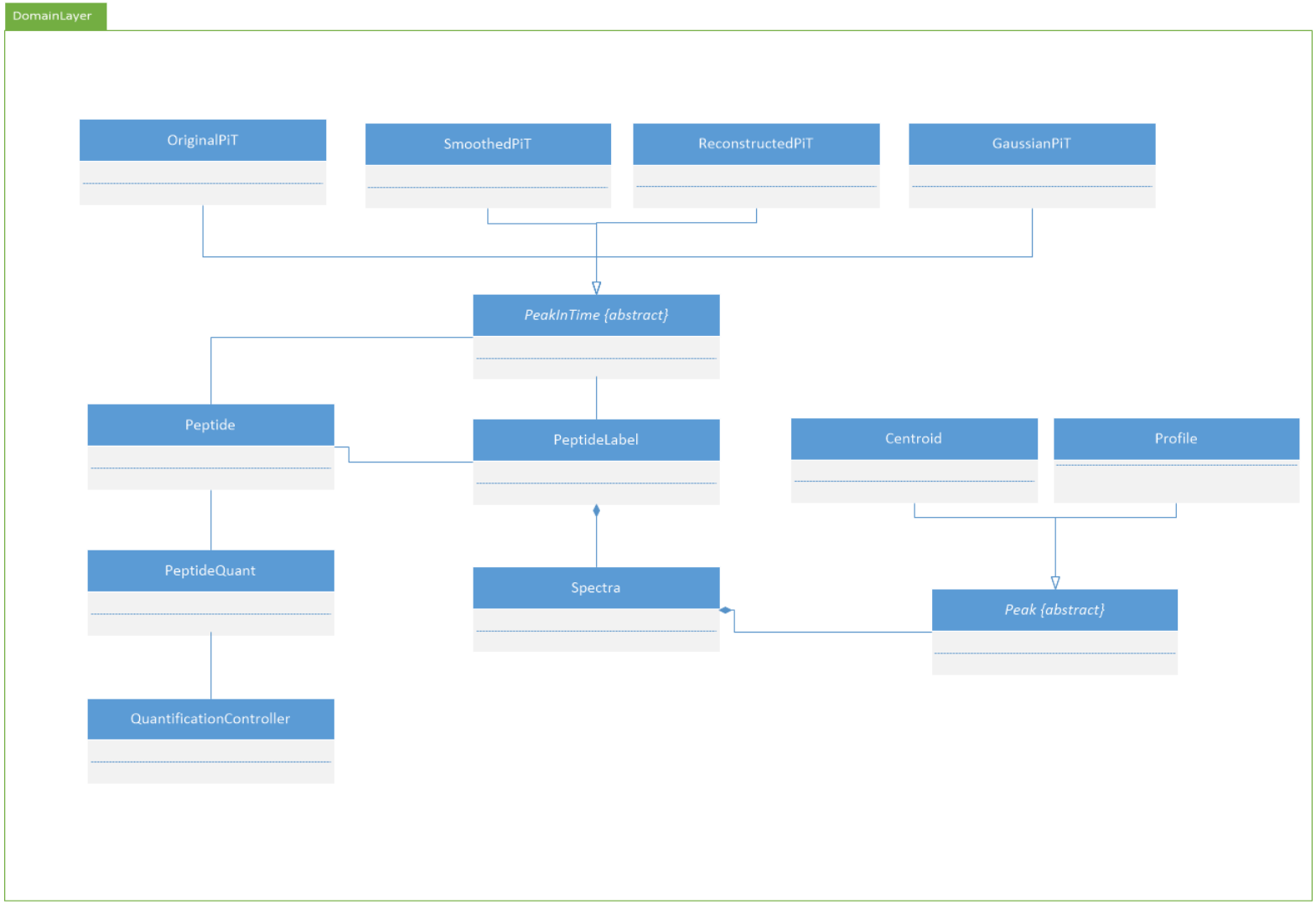


Figure 52: Proteomics quantification algorithm Domain Layer.

Figure 53: Proteomics quantification algorithm Auxiliar and Configuration packages.

4.4 Implementation decisions

The proteomics quantification algorithm has been implemented using the programming language JAVA.

> "Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and

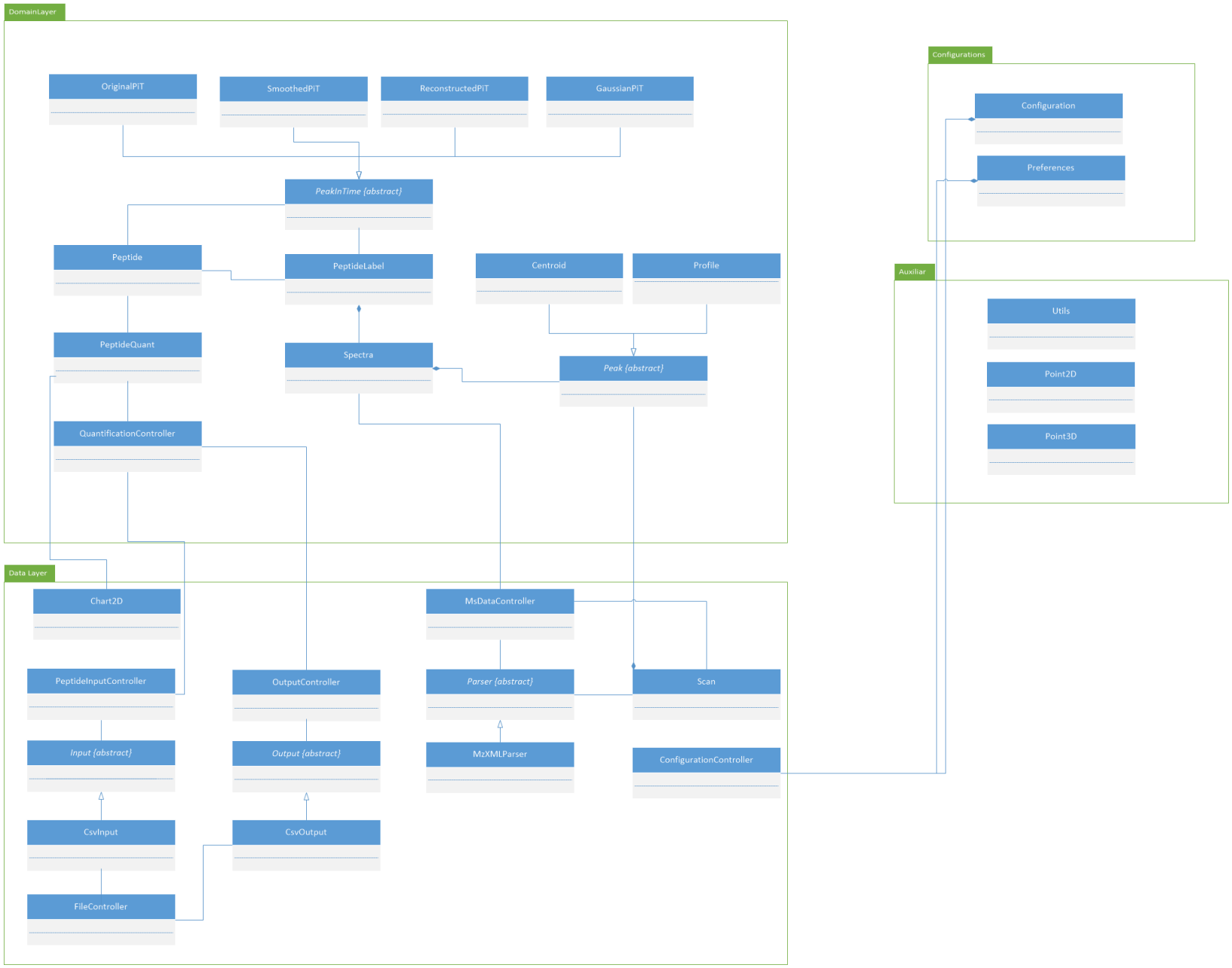


Figure 54: Proteomics quantification algorithm complete Class Diagram.

> reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!"

The main reason why it was used, is because JAVA is one of the most used and portable languages. The compiled algorithm can be used in multiple platforms without recompilation need, so that the algorithm can be easily shared, executed and used as a library. The other main reason is that the Villén laboratory is developing most of their applications in JAVA, so that the algorithm is easier to be used if it has been written in JAVA too.

Another implementation decision was to parallelize the code, this way the algorithm execution is faster than a serie execution. The parallelization has been done in the quantification permitting to quantify more than one peptide at time.

Finally the last implementation decision was to load all the needed mass spectrometer's output information into the algorithm own classes instead of consult the file every time that the algorithm needed information. This decision was made in order to improve the algorithm speed.

5 The proteomics quantification module

In order to use the proteomics quantification algorithm and view its results, a website module was created. This module is part of the Villén lab current platform but had to be easily portable to other new platforms.

In this section will be explained the user requirements for this module, shown their use cases, the module design, and finally explained the implementation decisions.

In this module only one stakeholder exists, the user. A user can be any genomics scientist that wants to use this system in order to quantify samples and watch their results, so that these two actions will be the module main goals.

5.1 Requirements

In this section all the module's functional and non-functional requirements will be explained.

5.1.1 Functional requirements

Functional requirements define the specific behavior or functions that the system must have.

This module has the following functional requirements:

Create configuration file The user should be able to create a configuration file for the proteomics quantification algorithm without knowing the file structure. An intuitive view should be shown to the user in order to complete the needed data: labels, dynamic and static modifications, method and isotope to be used for the area calculation, peptide treatments, MS noise window, mz noise window, peptide minimum peaks, cross quantification search peptide MS window, search peptide MS window, search peptide MS scans not found, isotopic distribution minimum peaks, max smooth times, use MS window, variable Mz, similar intensity, similar mz. All properties, except labels, dynamic and static modifications, will have a default value.

Load configuration file The user should be able to select any of the previously created configuration files in order to use it for quantification purposes.

Edit configuration file The user should be able to modify and save the changes of a previously created configuration file. The system will show the selected configuration file properties permitting to edit their values.

Batch quantification The user should be able to execute the quantification algorithm selecting the search of the run that wants to be quantified. The user should be able to indicate more than one run at once in order to put them into a quantification queue.

Cross quantification The user should be able to execute the cross quantification algorithm selecting the source and target searches of the runs that the user wants to cross quantify.

View quantification executions The user should be able to view all the quantified runs done with their quantification properties: search id of the run quantified, method used in order to calculate the area, isotope used for area calculation, peptide treatment done, and used configuration file.

View quantification results The user should be able to view for any quantified run and any of their peptides and their quantification information such as: label, noise, area, ratio, maximum intensity, scans, finding score...

Download quantification results The user should be able to download the selected quantification execution results in CSV format.

View peptide quantification information The user should be able to view the quantification information of any peptide previously quantified in a concrete quantification execution. The information that should be shown is:

- General information: execution id, run id, peptide search id, peptide id, peptide sequence, MS2 scan, peptide charge, peptide mz, peptide noise, algorithm used to calculate the area, isotope used for quantification, and previous area calculation peptide treatment.

- Labels info: every label will have: first scan, last scan, number of scans, score, area, maximum intensity, maximum intensity scan, maximum intensity retention time and its signal to noise value. The "non-light" labels will have the area and intensity ratios too.
- Images: At least will appear two images, the original elution chromatogram and the elution chromatogram obtained after the peptide treatment.

5.1.2 Non-functional requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

This module has the following non-functional requirements:

- The module will be secure: the users will only have access to their own runs, searches and quantification executions. Non-authenticated users will not have access to the module.
- The module will be easily usable, and the actions to be performed intuitive.
- The system can be used while the quantification algorithm is being executed.
- The module can be easily tested, maintained, scalable and extended.
- The module will be robust.

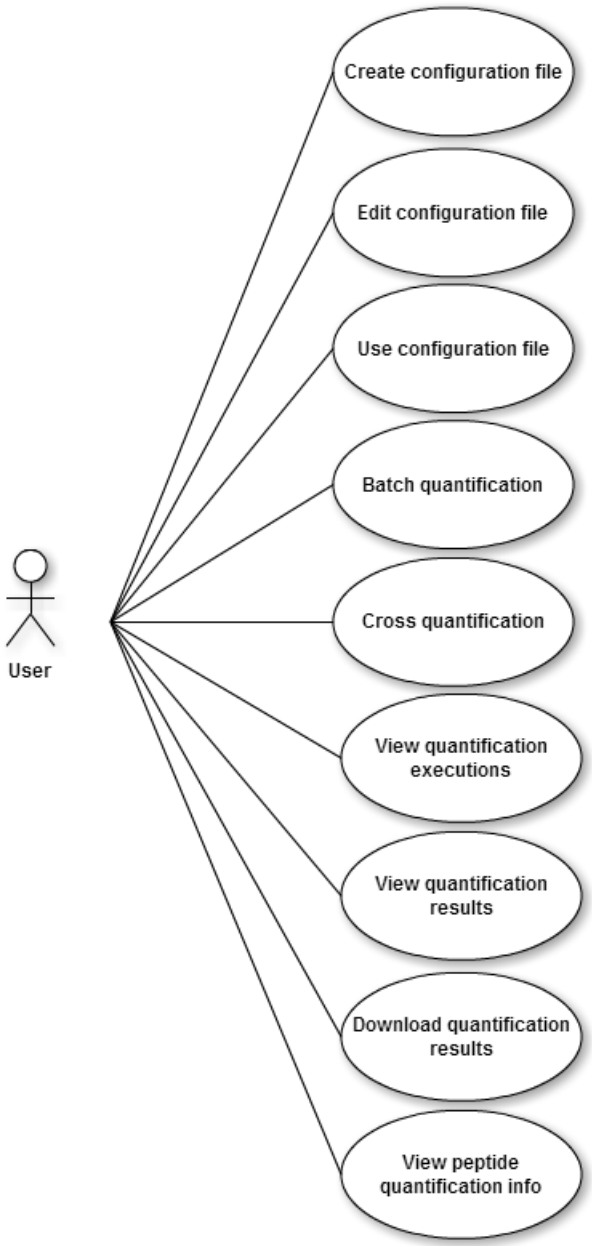


Figure 55: Proteomics quantification module Use Cases.

5.2 Design

In this section will be explained the module design, where will be detailed the different action flows.

The module is divided in two completely different actions: the quantification execution where can be decided the quantification action, selected, created or edited the configuration file, and finally performed the quantification; and the results view where can be seen or downloaded all the quantification executions with their peptides quantification data.

5.2.1 Quantification

In the quantification action, the first decision that must be made is decide if the user wants to perform a batch quantification or a cross quantification. If it is the batch quantification, the user must select all the searches that wants to be quantified; in case of selecting the cross quantification, the user must select the source and target searches to be cross quantified. In both cases, when the searches had been selected, the user can decide whether the configuration file should be created from zero or loaded. In case to be loaded, the user can decide to edit the selected configuration file and save the changes. Finally, when the configuration file parameters have been completed, the user can proceed to execute the quantification. In any case, the user can go back to the previous screen.

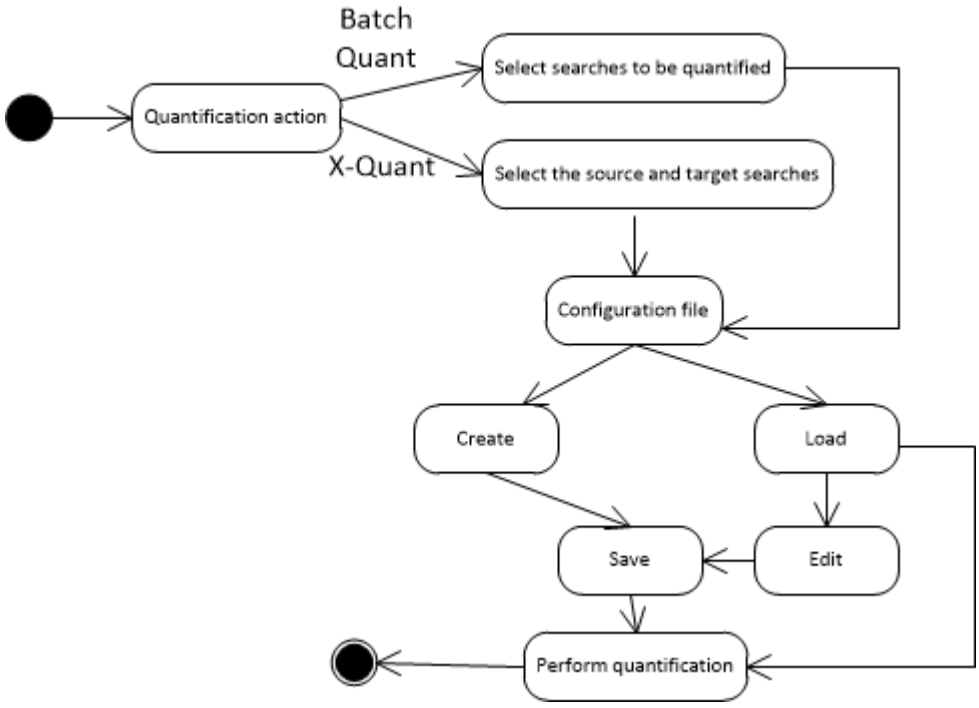


Figure 56: Quantification action flow.

5.2.2 Results view

In the quantification results view, all the performed quantification executions are shown. When the user selects one of them, its peptides quantification are shown permitting to the user download them or not. If the user selects a peptide, the concrete information relative to this peptide will be shown providing images of its elution chromatogram. In any case, the user can go back to the previous screen.

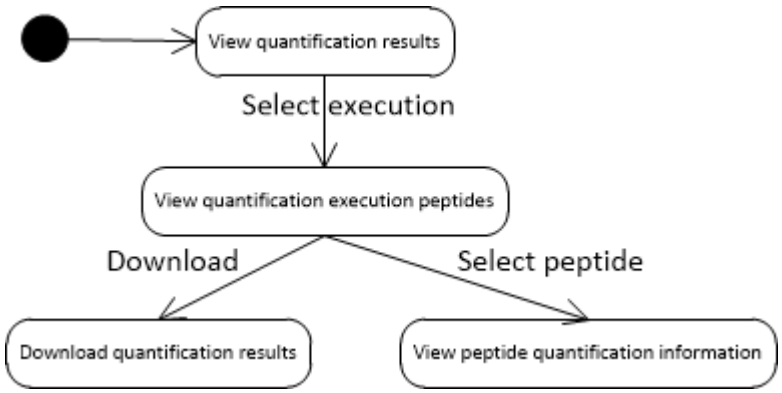


Figure 57: Quantification results view flow.

5.3 Implementation

Architecture
The module is divided in two different pages: the quantification page, and the results viewer.
This two pages are using a SOUI architecture (Stands for Service-Oriented User Interface), this means that they only have one page (view), served only once by the server; the different page views are controlled via JavaScript, hiding or showing page contents depending on user actions. The client only communicates to the server using asynchronous petitions (Ajax). This petition responses are in json format and only contains data (won't contain html).

Why SOUI? A SOUI architecture uses a clear separation between client and server. This way, the module can be deployed easily in different platforms, changing only the server code if is needed. Another advantage is that the page is only loaded once, the same client will show and hide content, obtaining this way faster responses. Another advantage is that the server load will be low because it only serves data.

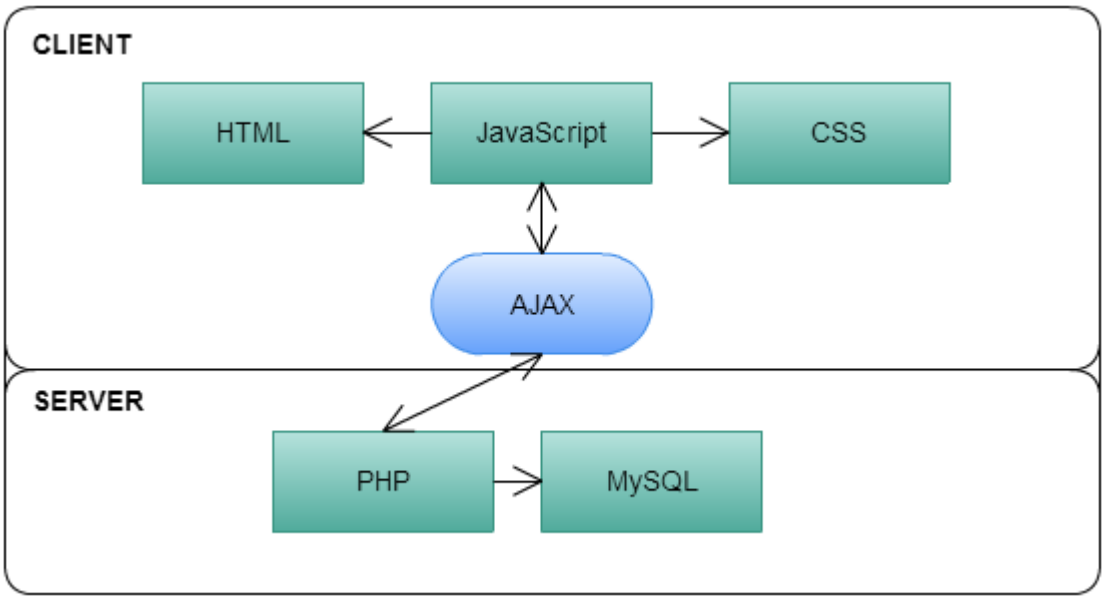


Figure 58: Client and server interactions via AJAX.

Structure
The module will have a clear program language separation between the client and the server code. The client code will provide the page views and user interactions (HTML, JavaScript) while the server will provide the petitioned data (PHP). This language separation will make a platform change easier, you can change the server code and/or programming language without having to change the client code. The runs and searches data are obtained from a MySQL database.

The module code is structured in different folders:

- css: Contains all the Cascade Style Sheets. The module is using Bootstrap 2.3.2 CSS [2].
- html: Contains the module two pages.
- img: Contains the Bootstrap 2.3.2 glyph icons.

[2]Global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system.

- js: JavaScript functions, this folder has a subfolder containing the jQuery [3] code and Bootstrap 2.3.2 JavaScript components. The module JavaScript code is divided for functionalities.

- php: Contains the php code. All the ajax petitions goes to a Front Controller named actionController.php, which depending on the petition received calls the appropriate function. This folder contains the module php code.

- cli: Contains the executable jar used for quantification and the needed php code to execute it.

- root folder: The code located in the root folder are the two main pages. These are module needs, the only thing that they do is include the corresponding HTML, and check if the user has permissions.

[3]jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

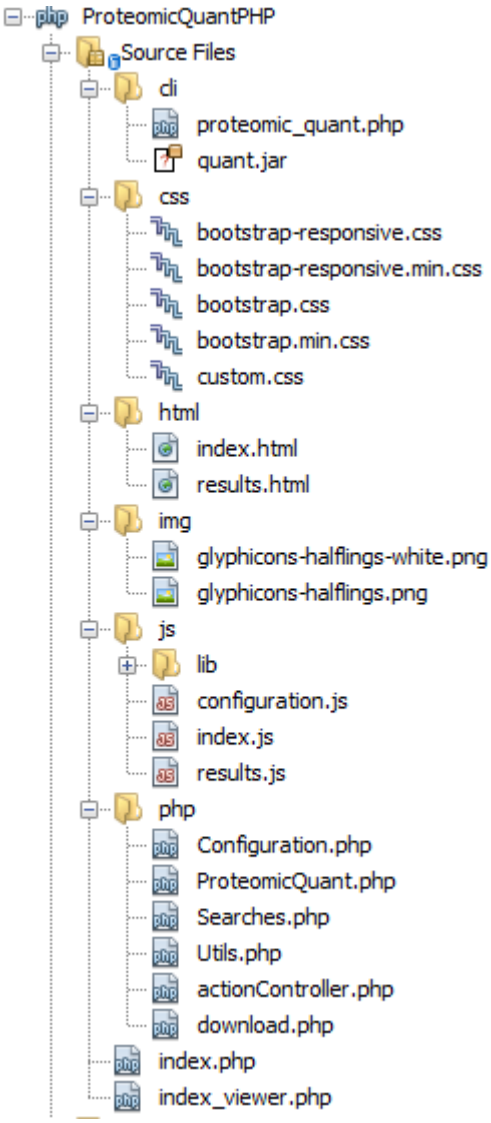


Figure 59: Module files structure.

In the next subsections will be explained the files of every action and shown the relation between them given a user interaction.

5.3.1 Quantification

The quantification action has its web view in the HTML index file. The different views contained within are shown or hidden via the index and configuration JavaScripts. The index JavaScript performs the quantification actions: select the quantification type and perform the quantification; while the configuration JavaScript controls the configuration file creation, loading and editing.
In order to perform the quantification, the available searches must be shown. These searches are loaded and validated via Ajax using the index JavaScript. In table 9 can be observed all the client to server interactions in order to provide and obtain data.

For quantification purposes, the actionController does interactions with two different classes: Searches.php and Configuration.php. The first mentioned PHP class obtains all the user searches and their related information, and checks if a given search is valid for the current user. The Configuration class, has the capability of create from zero a configuration file, read and send a configuration file parameters and save them.

5.3.2 Results view

The results quantification view has its web view in the HTML results file. The different views contained within are shown or hidden via the results JavaScript. This file does petitions in order to obtain the quantification executions data, and in case of select a peptide to view its information, performs a single quantification using the previously obtained encoded peaks. This quantification is really fast, so that the user does not have to wait more than 2 seconds.

For this purpose, the actionController does interaction with the ProteomicQuant class, which is capable of obtain any information related to a quantification execution. In case of wanting to download a quantification execution data, the actionController interacts with the ProteomicsQuant in order to prepare the CSV file with the quantification information, and the download.php file which proceeds to download the data.

It is important to note that in the module, the configuration file has been splitted in two different parts: one part that can be common to different executions, as can be the labels used, the amino acid modifications, the methods used for the area calculation, and the algorithm properties. And what could be named the dynamic part which are the input and output paths. Doing this separation the configuration files can be reused and in case of perform the single quantification the same algorithm values can be applied.

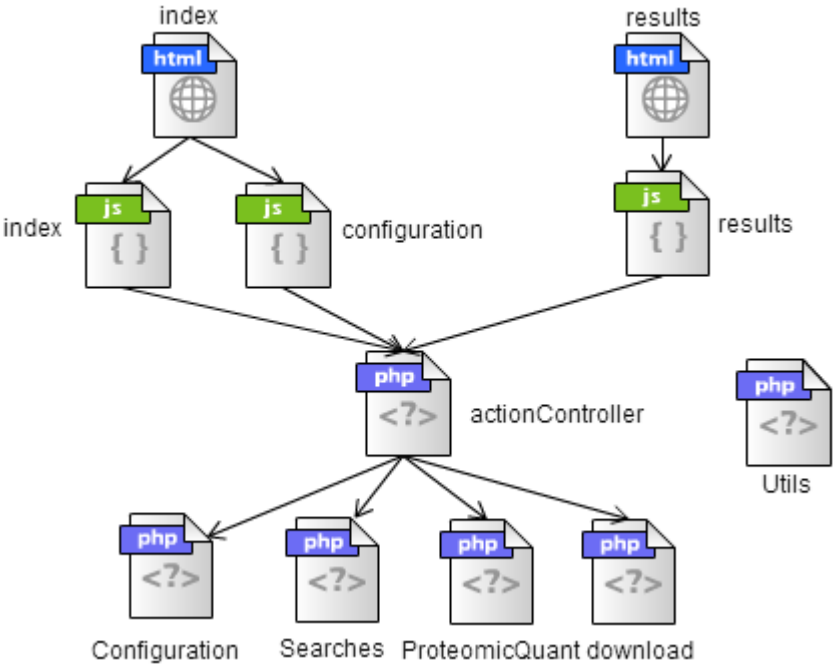


Figure 60: Module files interaction.

In this image can be seen the clear separation between the different module actions, and the client-server independence.

More information is explained on the annex module documents.

HTML	Action	JavaScript file	AJAX action	PHP class
index	Batch quantification	index	execute	ProteomicQuant
index	Cross quantification	index	execute	ProteomicQuant
index	Create configuration file	configuration	createCF	Configuration
index	Load configuration file	configuration	confFileData	Configuration
index	Edit configuration file	configuration	createCF	Configuration
index	Use configuration file	configuration	useCF	Configuration
index	Check if valid search id	index	checkSearchId	Searches
index	Get available searches to quantify	index	getInputs	Searches
index	Get stored configuration file names	configuration	loadConfigurationFiles	Configuration
results	View quantification executions	results	executions	ProteomicQuant
results	View quantification results	results	executionData	ProteomicQuant
results	Download quantification results	results	downloadExecDataId	ProteomicQuant and download
results	View peptide quantification information	results	peptideData	ProteomicQuant
results	Get a quantification execution search id	results	execSearchId	ProteomicQuant

Table 9: Module client-server interactions

6 Results analysis

In this section will be shown three different studies that were done: the algorithm ratios, the ppm error, and the peptide finding.

6.1 Algorithm

In order to see the algorithm accuracy, a study was done using prepared samples where the ratios were already known.

In the following table can be seen the runs and their corresponding ratios that were used in order to do the study.

Run	Type	Ratio (log2)
3218	1-1	0
3219	1-2	1
3220	1-4	2
3221	1-8	3
3222	1-16	4
3223	2-1	-1
3224	4-1	-2
3225	8-1	-3
3226	16-1	-4

Table 10: Runs types and ratios

The results provided by the explained algorithm should have the same ratio. The ratio is decided using the run ratio median because of its robustness.

Lets see the comportement:
As can be seen in figure 61 all methods set the correct ratios to the runs. Every run had the same ratio that their known one. Of course, some methodologies worked better than others. The best methodologies used are all the PPM Score variants and the Smooth method calculating the area using the gaussian method.

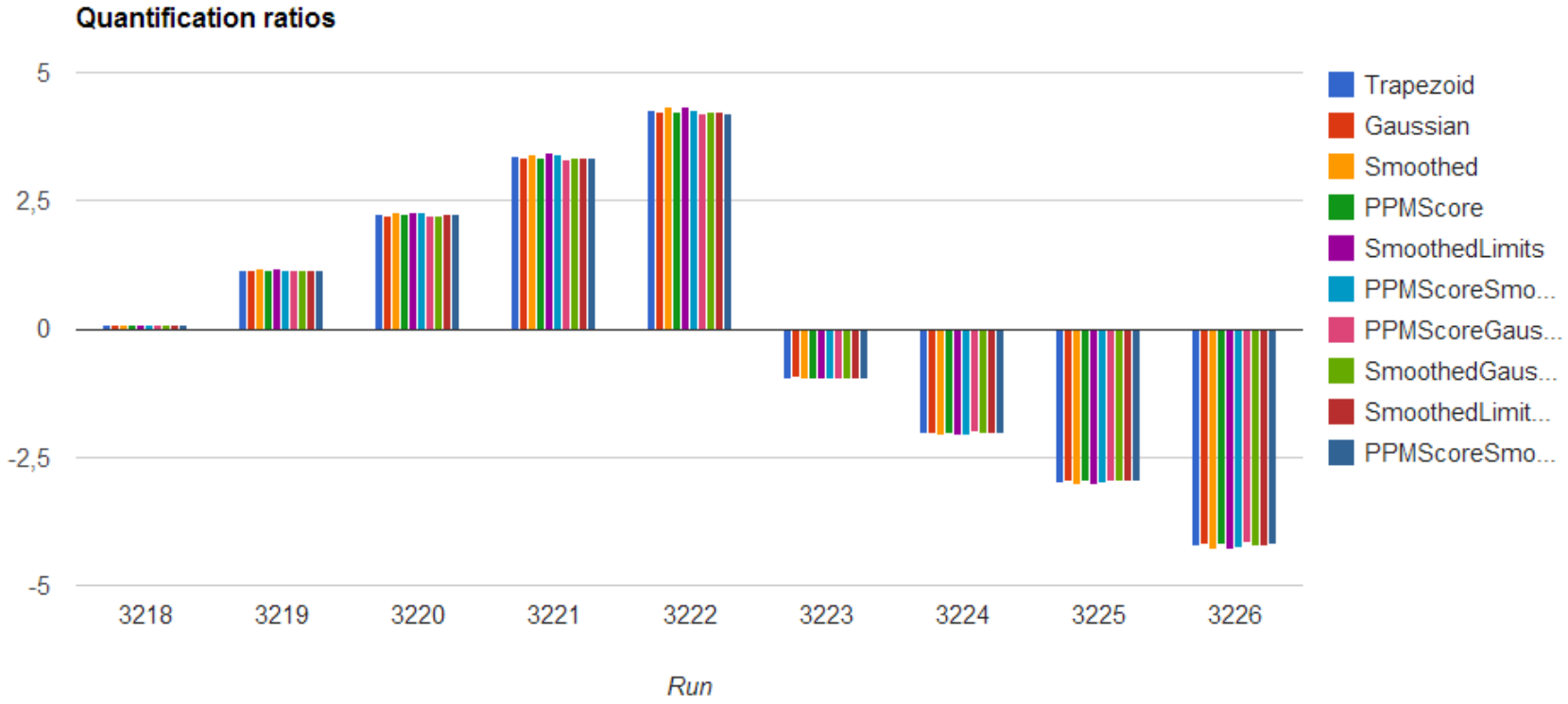


Figure 61: Quantification runs ratio median. This figure shows the ratios provided by the different algorithm methodologies.

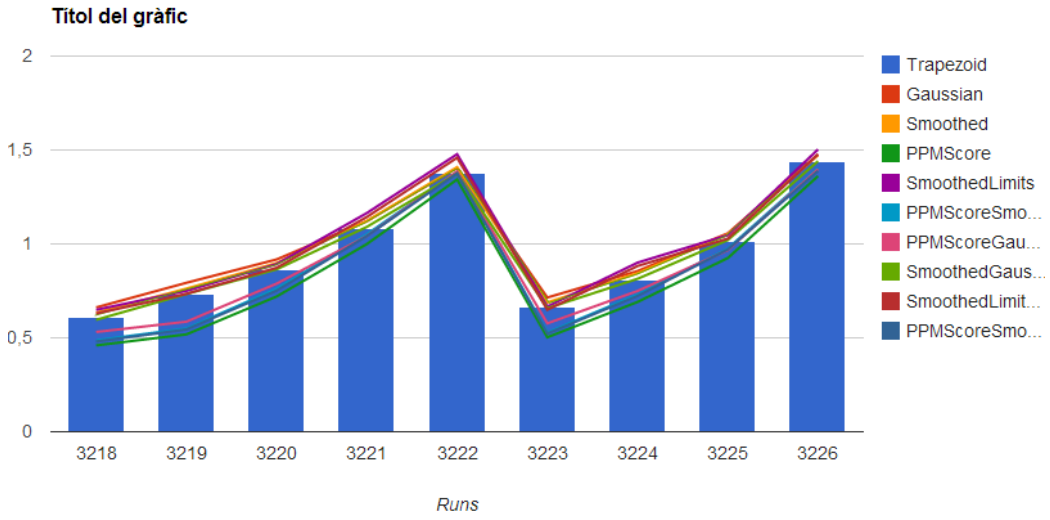


Figure 62: Quantification runs ratio standard deviation. As bigger is the expected ratio, bigger is the standard deviation.

As can be seen as bigger is the ratio bigger is the standard deviation. This is a natural conclusion because as bigger is the distance between labels bigger is the accumulated noise within it, and for that, the error associated.

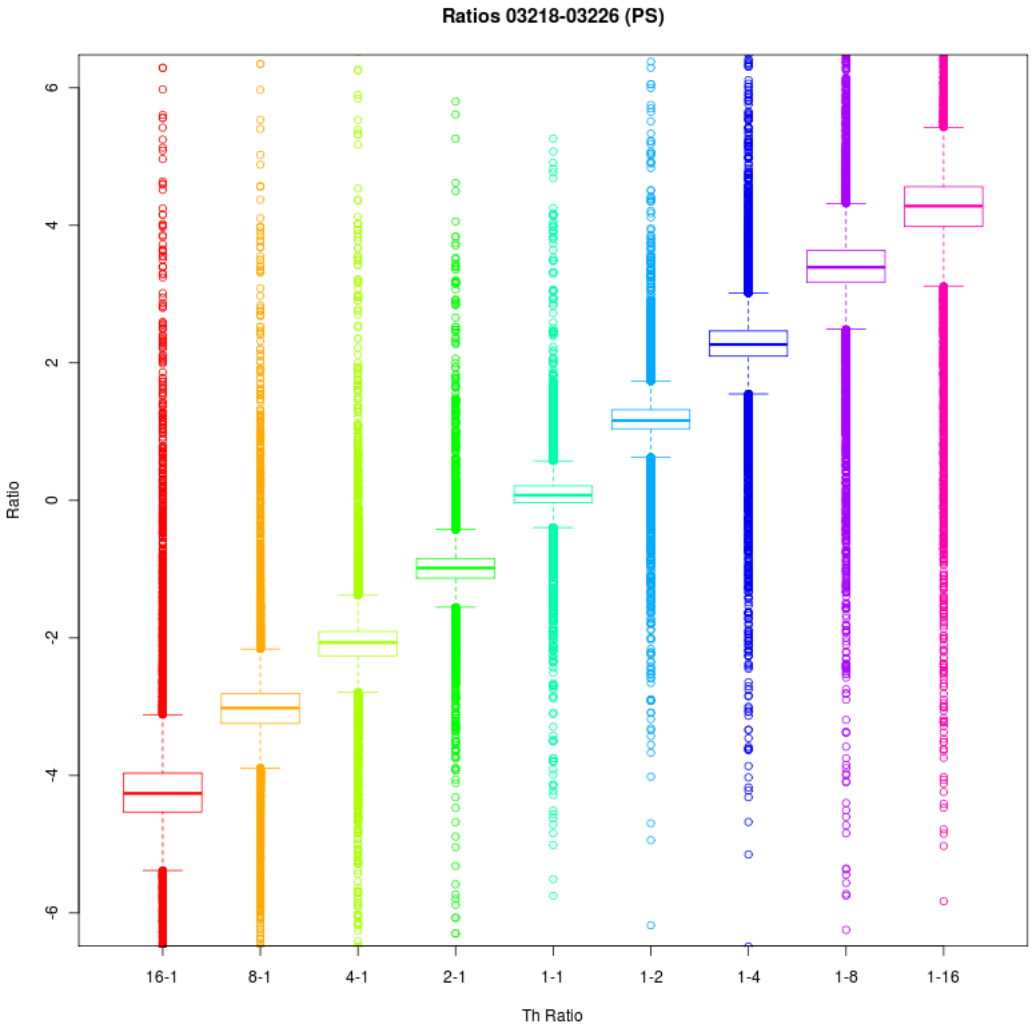


Figure 63: Quantification runs ratios boxplots. These ratios were calculated using the PPM Score peptide treatment and the smooth after that.

6.2 PPM study

A study was done in order to calculate the most appropriate ppm error for the peptide's mz search. To do so, a set of runs were used in order to obtain the theoretical mz and the real mz that the peptides were found with. Calculating their difference, the ppm errors were obtained, which were

plotted in order to see the better ppm error to be used.
As can be observed in the below image the most part of the peptides are contained within the +/- 3 ppm. There are a few outliers outside this window, but this ppm window is a good fitting. Any ppm error between 3 and 5 should be well considered.

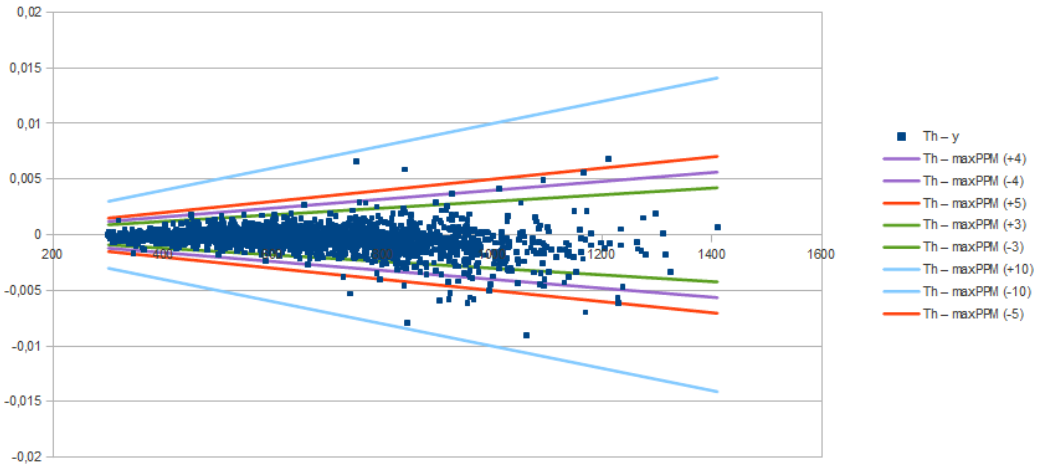


Figure 64: PPM study showing the existing ppm error between the theoretical mz and the found ones.

6.3 Peptides finding

This study shows the percentage of the total peptides of a run that were found and quantified in the proteomics quantification algorithm. This study was done searching the peptide with different number of isotopes: the monoisotopic isotope, the first isotope, and the second.
In the table shown in figure 65 can be clearly seen that as smaller is the peptide more difficult is to find it, this is a natural conclusion because as smaller is the peptide more probable will be to the mass spectrometer to consider it noise. An example could be a 1-4 ratio, where in this case the Light (1) will be smaller than the Heavy (4), and for this reason less probable to be found 0.69 vs 0.85.
The images shown in figure 66 show the explained tendency, as bigger is the ratio between the peptides less peptides of the smaller label will be found because will be more similar to the noise. In the ratios where the Light label is the smaller one, the percentage of findings is smaller as bigger is the

	Light – 0	Heavy – 0	Total – 0	Light – 1	Heavy – 1	Total – 1	Light – 2	Heavy – 2	Total – 2
1-1	0,86	0,84	0,83	0,86	0,85	0,83	0,81	0,78	0,75
1-2	0,82	0,85	0,80	0,82	0,85	0,80	0,74	0,81	0,71
1-4	0,69	0,85	0,67	0,70	0,85	0,68	0,59	0,80	0,57
1-8	0,50	0,85	0,48	0,51	0,85	0,49	0,42	0,80	0,40
1-16	0,36	0,85	0,34	0,36	0,85	0,34	0,29	0,79	0,27
2-1	0,87	0,81	0,81	0,87	0,81	0,81	0,83	0,71	0,70
4-1	0,87	0,69	0,68	0,87	0,69	0,69	0,83	0,57	0,56
8-1	0,87	0,53	0,52	0,87	0,53	0,52	0,83	0,42	0,41
16-1	0,87	0,32	0,32	0,87	0,33	0,32	0,82	0,25	0,25

Figure 65: This table indicates for every run the percentage of the peptides that were found for every label (Light and Heavy) and in total (both cases). This study is done using the monoisotopic - only one isotope (yellow), two isotopes - until the 1st isotope (pink), and three isotopes - until the 2nd isotope (blue).

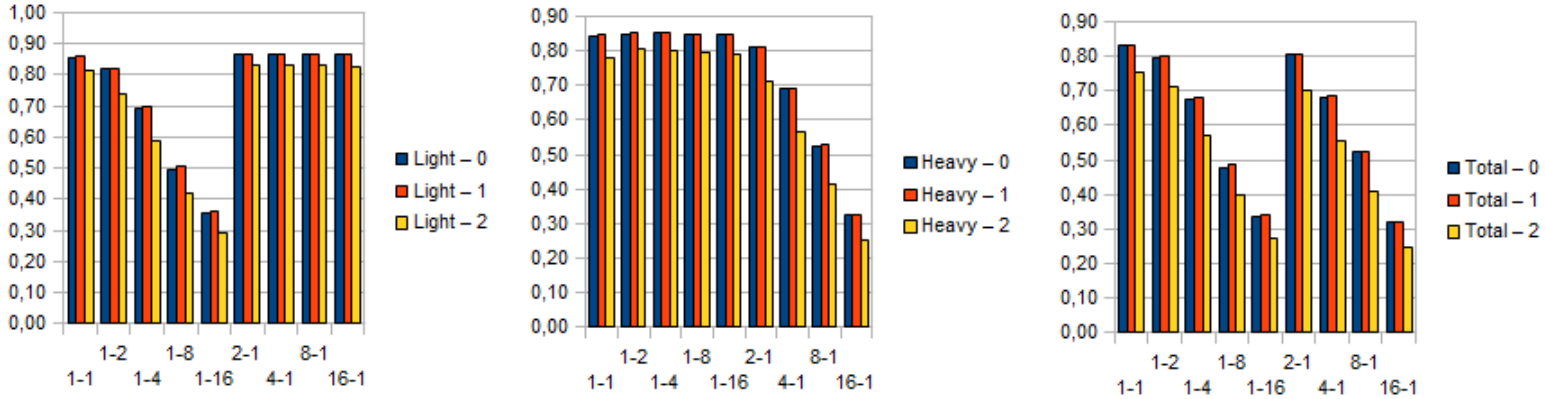


Figure 66: The first image shows the light percentage of findings for every different ratio, the second shows the heavy ones, and finally the third image shows the percentage of peptides where both labels were found. For every run is represented the percentage of peptides found finding one isotope (blue), two (red), or three (yellow).

ratio difference with the Heavy one (ratios 1-x). The same happens with the Heavy label, in the ratios where it have the small peptides, the percentage of findings is smaller as bigger is the ratio difference with the Light ones (ratios x-1). In conclusion, as bigger is the ratio between peptides, less peptides will be found in order to perform the quantification.

Another conclusion that can be made is that the algorithm works better if it is using two isotopes to be found.

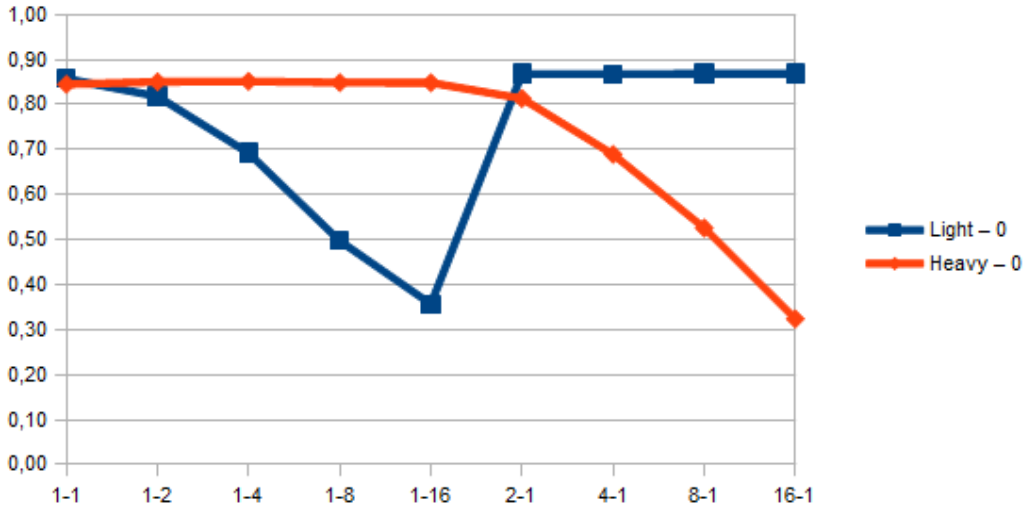


Figure 67: Image showing how the light (blue) and heavy (red) finding decreases when the ratios are bigger.

7 Future improvements

The quantification algorithm can be improved in its different sub-algorithms in order to obtain better results or provide more flexibility to the user:

- The peptide search algorithm could be improved adding a new option in order to auto-calculate the acceptable ppm variance for the current run.
- New peptide treatments could be added into the algorithm in order to obtain a better peak in time shape.
- New methods for area calculation could be added into the quantification algorithm.
- Different input and output formats could be added into the system in order to be more flexible in the quantification algorithm. One of these new formats could be pepXML.
- In the cross quantification algorithm a new method could be added in order to calculate the target scan equivalent to the source one.
- The module could show the calculated linear regression in the cross quantification execution in order to provide to the user a better understanding of the obtained results.

8 Alternatives

Nowadays exists different quantification softwares that could be used for quantifying purposes, for example: MaxQuant, PEAKS, and Vista.

MaxQuant
MaxQuant is a quantitative proteomics software package designed for analyzing large mass-spectrometric data sets. It is specifically aimed at high-resolution MS data. Several labeling techniques as well as label-free quantification are supported by the software. MaxQuant is freely available and can be downloaded from their site.
A MaxQuant inconvenient is that is not portable, it is only available for Windows. Protein quantification can be performed in three ways in this software: using (a) only unique peptides (b) unique peptides and razor peptides (c) all peptides. Using all peptides is usually not recommended, and this is an important feature that the quantification algorithm should have. But the most important inconvenient is that this quantification software can not be modified, for this reason, new features and/or preferences can not be added.

PEAKS 7
PEAKS is a complete software package for proteomics mass spectrometry data analysis. Starting from the raw mass spectrometry data, PEAKS effectively performs peptide and protein identification, PTM and mutation characterization, quantification (label and label free) as well as result validation, visualization and reporting. The most important inconvenient that this software has is that this software needs a license to be used and can not be modified for the laboratory actual and future needs.

Vista
Vista is a software suite for the automated analysis of relative abundance ratios from MS-based proteomic data developed by Corey Bakalarski in the Gygi Laboratory at Harvard Medical School. It works with a variety of stable isotope labeling methods, and provides quick, easy calculation of thousands of abundance ratios per hour. It was specifically designed to work with data with highly-accurate masses, and incorporates a number of novel improvements to existing tools. This software works through a convenient web-based

system - just install and use from anywhere you have an internet connection. Vista is a privative software and they are currently working on a academic use only version. For this reason, Vista is not an option for the laboratory.

9 Economical analysis

An economical analysis has been done in order to know the costs of the proteomics quantification algorithm and its module development.

Hardware
In order to develop the project, the laboratory provided a semi-new AMD computer with 16GB of RAM. In order to know the representing cost of the computer for the project development, the time that was used for this purpose will be calculated and obtained the percentage of its life expectancy. So that the cost for this project will be the corresponding percentage of its original cost, this is because the computer was semi-new.
The life expectancy for a computer is of 4 years, and the time that the computer was used for this project was 6 months, so that the usage percentage is of 12.5%

Hardware	Price	Project price
Computer (AMD, 16GB of RAM)	1000€	125€

Table 11: Project hardware costs

Software
All the software used in order to develop the project was free: the operative system used was Ubuntu, the algorithm was written using Eclipse, and the module was developed using Netbeans. All these software were completely free, so that the software cost was 0€.

Staff
In order to develop the project 3 people with different roles participated: the project manager, the laboratory computer scientist and the project programmer. The project manager checked the project 1 hour per week, the laboratory computer scientist 2 hours per week, and the project programmer worked on this project 8 hours per day. The project was developed in 6 months (laboral days).
The staff total cost was 41280€.

Staff	Price/Hour	Total hours	Price
Project manager	70€/h	24	1680€
Computer scientist	55€/h	48	2640€
Project programmer	35€/h	1056	36960€

Table 12: Project staff costs

The final project cost was 41405€.

10 Conclusion

The proteomics quantification algorithm and the web based module that uses the algorithm could be developed within the internship time. This means, that from that time, all the laboratory scientists could use the proteomics quantification algorithm (nowadays named Thunder Quant) for their investigations in their daily works.

Target	Sub-target	Achieved	Additional features
Algorithm	Obtain peak in time	✓	Personalized search
	Isotopic distribution abundance	✓	
	Calculate noise	✓	Personalized windows
	Peptide treatment	✓	Smooth, Reconstruction
	Area calculation	✓	Gaussian function
			Cross quantification
			Single quantification
			Configuration file
Module	Execute	✓	
	View results	✓	
	Download results	✓	
			Configuration file editor

Table 13: Achieved project targets and additional features implemented.

On table 13 can be seen that all the project targets defined beforehand were successfully developed, obtaining a 100% of accomplishment. Moreover, some other features could be added to the algorithm and module in order to provide better results and a higher usability.

The proteomics quantification algorithm is completely modular and can be expanded easily without complications. The algorithm, or parts of it, can be used as a library of bigger algorithm thanks to its high portability.

Thanks to the proteomics quantification module client side independency of the server language, the migration into the new platform should be done easily, fast, and without complications.

Both implementations (algorithm and module) are solid structures that can be expanded and improved along the time.

As a personal conclusions, I consider that this project developement gave me new and improved knowledges of mass spectrometry data treatment, web development, parallelism... Moreover, as I worked in a multidisciplinary laboratory, I was able to learn different concepts related to biology and bioinformatics from this project and other project meetings, providing to me a very enriching experience.

11 Glossary

Molecule electrically neutral group of two or more atoms held together by covalent chemical bonds. Molecules are distinguished from ions by their lack of electrical charge.

Protein large biological molecules consisting of one or more chains of amino acids. Proteins perform a vast array of functions within living organisms, including catalyzing metabolic reactions, replicating DNA, responding to stimuli, and transporting molecules from one location to another.

Peptide short chains of amino acid monomers linked by peptide (amide) bonds. Peptides are distinguished from proteins on the basis of size.

Amino acid are biologically important organic compounds made from amine (-NH2) and carboxylic acid (-COOH) functional groups, along with a side-chain specific to each amino acid. The key elements of an amino acid are carbon, hydrogen, oxygen, and nitrogen, though other elements are found in the side-chains of certain amino acids. They are represented by a uppercase letter.

Atom basic unit of matter that consists of a dense central nucleus surrounded by a cloud of negatively charged electrons.

Isotope variants of a particular chemical element: while all isotopes of a given element share the same number of protons and electrons, each isotope differs from the others in its number of neutrons.

Ion atom or molecule in which the total number of electrons is not equal to the total number of protons, giving the atom a net positive or negative electrical charge.

Ionize convert (an atom, molecule, or substance) into an ion or ions, typically by removing one or more electrons.

Electron subatomic particle with a negative elementary electric charge.

Neutron subatomic hadron particle which has the symbol n or n0, no net electric charge and a mass slightly larger than that of a proton.

Proton subatomic particle with the symbol p or p+ and a positive electric charge of 1 elementary charge.

Carbon chemical element with symbol C and atomic number 6. As a member of group 14 on the periodic table, it is nonmetallic and tetravalent—making four electrons available to form covalent chemical bonds.

Isotopic distribution can be understood as a distribution of peaks that has $(100 - i)$ Carbon 12 atoms and i Carbon 13 atoms, where i is the isotope position in the distribution starting by 0.

Charge (z) a fundamental physical attribute of a particle, which characterizes the particleselectromagnetic interaction with other particles and with electric and magnetic fields.

Mass spectrometry is the art of measuring atoms and molecules to determine their molecular weight. Such mass or weight information is sometimes sufficient, frequently necessary, and always useful in determining the identity of a species. Mass spectrometry (MS) is an analytical technique that produces spectra (singular spectrum) of the masses of the atoms or molecules comprising a sample of material. The spectra are used to determine the elemental or isotopic signature of a sample, the masses of particles and of molecules, and to elucidate the chemical structures of molecules, such as peptides and other chemical compounds. Mass spectrometry works by ionizing chemical compounds to generate charged molecules or molecule fragments and measuring their mass-to-charge ratios.

Label is a mass spectrometry strategy used in quantitative proteomics.Peptides or proteins are labeled with various chemical groups in order to identify a sample in the spectra.

Peak 1. mz signal in a given retention time. 2. bell shape of an elution chromatography.

Mz the importance of the mass-to-charge ratio, according to classical electrodynamics, is that two particles with the same mass-to-charge ratio move in the same path in a vacuum when subjected to the same electric and magnetic fields

Abundance The abundance of a chemical element measures how relatively common (or rare) the element is, or how much of the element is present in a given environment by comparison to all other elements.

Ratio In layman's terms a ratio represents, for every amount of one thing, how much there is of another thing.

Spectra or mass spectrum A mass spectrum is an intensity vs. m/z (mass-to-charge ratio) plot representing a chemical analysis.[1]Hence, the mass spectrum of a sample is a pattern representing the distribution of ions by mass (more correctly: mass-to-charge ratio) in a sample.

Scan Mass spectrometry set of isotopes detected in a concrete retention time.

MS1 scan Mass spectrometry level one scans. This scan contains the ions that has been detected in its retention time.

MS2 scan Mass spectrometry level two scans. Precursor scan that indicates that an ion has been detected (has fallen) in its retention time. This scan has more concrete information relative to this ion.

Retention time is the characteristic time it takes for a particular analyte to pass through the system (from the column inlet to the detector) under set conditions.

Peak in time Set of mz peaks sorted by the retention time where were detected.

Elution chromatogram In this project refers to a graphical peptide peak in time for.

ppm the parts-per notation is a set of pseudo units to describe small values of miscellaneousdimensionless quantities. In this project mz ppm.

FWHM Full width at half maximum of a gaussian bell (at maxintensity/2 height).

FWTM Full width at tenth to minimum of a gaussian bell (at maxintensity/10 height).

12 Bibliography

References

[1] Corey E. Bakalarski, Joshua E. Elias, Judit Villén, Wilhelm Haas, Scott A. Gerber, Patrick A. Everley, and Steven P. Gygi: The Impact of Peptide Abundance and Dynamic Range on Stable-Isotope-Based Quantitative Proteomic Analyses. Department of Cell Biology, Harvard Medical School, Boston, Massachusetts 02115. Journal of proteome research. April 30, 2008.

[2] Jürgen Cox and Matthias Mann: MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. Nature biotechnology volume 26, December 2008.

[3] Johannes Griss1, and Christopher Gerner: GPDE: A Biological View on PRIDE. Journal of Proteomics and Bioinformatics, April 2009.

[4] Scott E. Van Bramer: An Introduction to Mass Spectrometry. Widener University, September 2, 1998.

[5] R. Martin Smith: Understanding Mass Spectra. A Basic Approach. Wiley, 2nd edition 2004.

[6] Victor P. Andreev, Lingyun Li, and Barry L. Karger: A New Algorithm Using Cross-Assignment for Label-Free Quantitation with LC/LTQ-FT MS. Journal of Proteome Research. June 2007.

[7] Alison Gibbs and Jeffrey Rosenthal. 'Making Sense of Data'. University of Toronto. 2013.
`https://class.coursera.org/introstats-001/lecture`

[8] Jeff Sauro. 'Measuring Usability'. 2013.
`http://www.measuringusability.com/zcalc.htm`

[9] Bruce Simmons. 'Mind on Statistics, 4th edition'.

[10] 'Stat Trek'. 2013.
`http://stattrek.com/regression/linear-regression.aspx#`

ReqressionPrerequisites
http://stattrek.com/regression/slope-confidence-interval.aspx

[11] Rishi Verma. 2013. 'Rishi Software'. January 24, 2012.
http://rishisoftwareblog.wordpress.com/2012/01/24/computing-t-distribution-cumulative-probability-density-function/

[12] Appache Commons (Simple Regression). 2013.
http://commons.apache.org/proper/commons-math/userguide/stat.html#a1.4_Simple_regression

[13] Wikipedia. 2013.
z-Score:
http://en.wikipedia.org/wiki/Standard_score
Confidence interval:
http://en.wikipedia.org/wiki/Confidence_interval
Gaussian function:
http://en.wikipedia.org/wiki/Gaussian_function Mass spectrometry:
http://en.wikipedia.org/wiki/Mass_spectrometry

[14] NASA/IPAC Extragalactic Database (NED). 2013.
http://ned.ipac.caltech.edu/level5/Leo/Stats2_3.html

[15] JFreeChart. 2013.
http://www.jfree.org/jfreechart/

[16] UC Davis. 'ChemWiki'. 2013.
http://chemwiki.ucdavis.edu/Analytical_Chemistry/Instrumental_Analysis/Mass_Spectrometry/Mass_Spectrometry%3A_Isotope_Effects

[17] Lamond Laboratory. 'The cell biologist's guide to proteomics'. 2013.
http://www.lamondlab.com/MSResource/LCMS/MassSpectrometry/massSpectrometry.php

[18] Sashimi. 'How to write an mzXML 2.1 conformant file'. 2013.
http://sashimi.sourceforge.net/schema_revision/mzXML_2.1/Doc/mzXML_2.1_tutorial.pdf

[19] Flatdog Media. 'Professional Surveyvor' (PPM). 2013.
http://www.profsurv.com/magazine/article.aspx?i=1196

[20] Jena, 'Lehrstuhl Bioinformatik'. 2013.
http://bio.informatik.uni-jena.de/wp/wp-content/uploads/2012/07/book_cms_chapter4.pdf

[21] 'Isotope Peaks of Ionic Fragments in Mass Spectrometry'. 2013.
http://www.chem.uoa.gr/applets/AppletMS/Appl_Ms2.html

[22] Code Strategies. 'AVAJAVA' (Threads). 2013.
http://www.avajava.com/tutorials/lessons/how-do-i-use-threads-join-method.html

[23] Oracle and/or its affiliates. 'Oracle' (Property file). 2013
http://docs.oracle.com/javase/6/docs/api/java/util/Properties.html

[24] 'Philosophy to Chemistry to Elucidation'. 2013.
http://acdlabs.typepad.com/elucidation

[25] 'Bioestadística: Métodos y Aplicaciones'. Universidad de Málaga. 2013
http://www.bioestadistica.uma.es/libro/node42.htm

[26] Isotope distributions. Freie Universität Berlin. 2013.
http://www.mi.fu-berlin.de/wiki/pub/ABI/QuantProtP4/isotope-distribution.pdf

13 Annex

13.1 Algorithm user manual

The next manual explains how can be executed the quantification software, having a special emphasis on how the inputs should be structured.

Quantification software

User Manual

Introduction

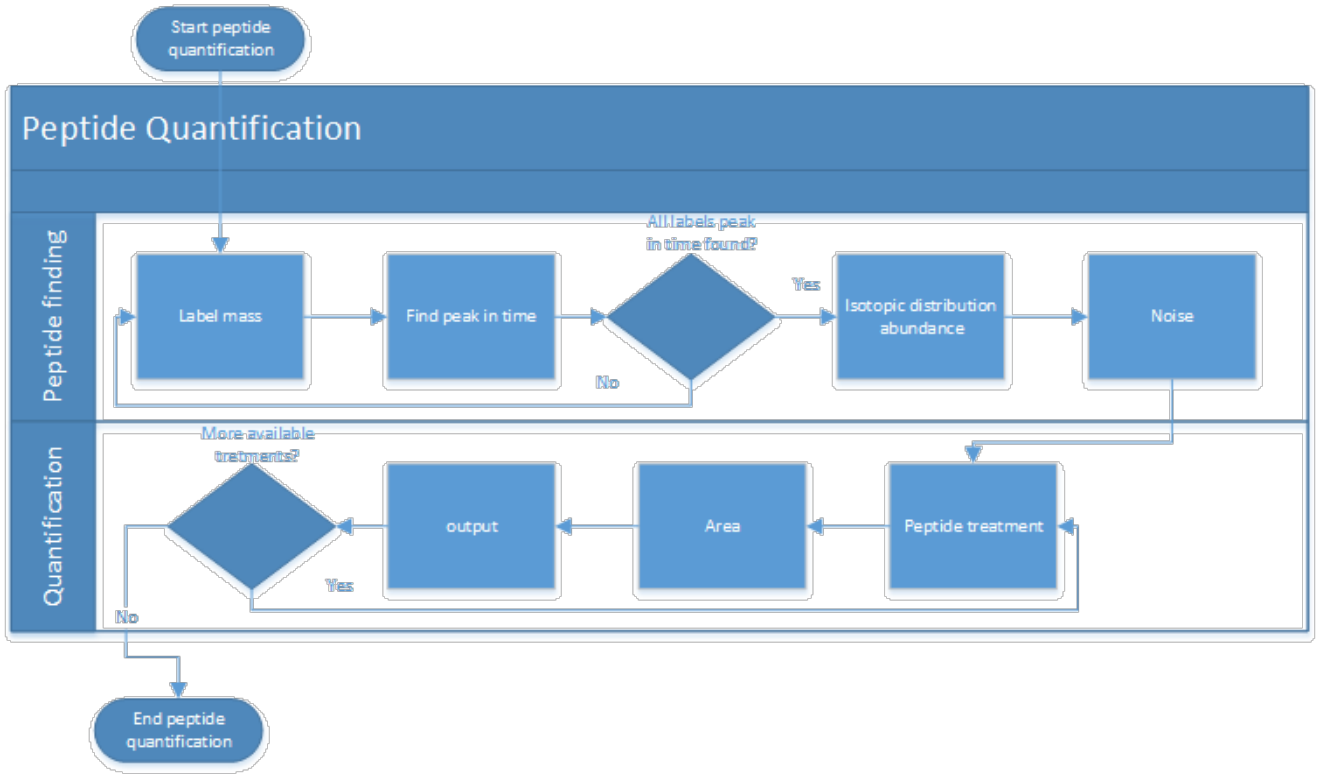
The proteomic quantification software is an executable jar that can be ran providing only a configuration file. This way, the algorithm can be easily used as a library or module.

This document will start explaining the algorithm basics, will continue showing how to create the software configuration file, and will end showing how can be executed.

The algorithm basics

The peptide quantification algorithm can be divided in two parts: the peptide finding and the quantification itself.

The peptide finding contains the peptide labels search and checks, while the quantification does the needed peptide treatment and area calculation.



In the above diagram is shown all the required steps for peptide finding and quantification.
The flow starts when a peptide is required to quantify; the first action will be to calculate the first label peptide mass and find its peak in time using this obtained mass. These steps will be repeated until all labels are found. Once the algorithm has all labels peak in time, will proceed to check if there are isotopic distribution abundance overlappings and if it's the case, will solve them. Finally, the peptide finding last step will be the noise calculation. Finished the peptide finding, the algorithm will start the quantification execution: the first thing that will be done is the peptide treatment, continuing with the area calculation, and the output data storing. This quantification steps will be repeated for every peptide treatment enabled and will be seen as a different output.

For further information check the *algorithm basics* document.

Configuration file

The quantification software configuration file contains all the needed information in order to run the algorithm.

Format file

Reads a property list (key and element pairs) from the given input path.
Properties are processed in terms of lines. A line is defined as a line of characters that is terminated either by a set of line terminator characters (\n, \r or \r\n) or by the end of the stream. A natural line may be either a blank line or a comment line.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#', '[' or ';' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab ('\t'), and form feed ('\f') to be white space.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', or white space character other than a line terminator. Any white space after the key is skipped; if the first non-white space character after the key is '=', then it is ignored and any white space characters after it are also skipped. All remaining characters found before a ';' on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". A ';' character is seen as a comment indicator; all text written in the line after this character will be ignored.

Lets see a correct configuration file example:

```
[SINGLE QUANTIFICATION]
inputPeptidesFilePath = /path/file.ext
inputPeptideFileFormat=/path/file.ext

[MS]
ppm   =10          ; ppm tolerance: 10 is +/- 10
#msWindow
 minPeaks=  5
      !maxSmooth = 4
```

From this configuration file, we will obtain:

```
"inputPeptidesFilePath" → "/path/file.ext"
"inputPeptideFileFormat" → "/path/file.ext"
"ppm" → "10"
"minPeaks" → "5"
```

All white lines, white spaces and comments were successfully skipped.

Configuration file properties

The configuration file properties provide to the algorithm its needed information; as the algorithm action, the input and output paths, the mass modifications, execution preferences...

- Action
This property indicates the algorithm action to be executed. The property will be identified by the program.action key and its value will be `QUANT` for batch quantification, `XQUANT` for cross quantification, or `SQUANT` for a single quantification.

Example:
```
program.action = XQUANT
```

- Inputs
The input properties vary depending on the wanted action:

For a `QUANT` action, two different input files must be provided: the peptides search result file path and the MS file path. This properties will be represented by the `inputPeptidesFilePath` and `inputMsFilePath` keys and their value will be the files path.

Example:
```
inputPeptidesFilePath = /runs/Ratios2labels/03218.csv
inputMsFilePath =  /share/Ratios2labels/03218.mzXML
```

The search file must contain the columns: peptide sequence, MS2 scan, peptide charge, peptide id, run id, search id, and theoretical mz.

For a `XQUANT` action, at least four input files must be provided. The cross quantification has two different inputs: sources and targets. Source inputs contains the peptides to be searched in the target inputs. For every source and target, a peptides search result file path and a MS file path must be provided. This properties will be represented by `xQuantSourcePeptidesFilePath_NUM`, `xQuantSourceMsFilePath_NUM`, `xQuantTargetPeptidesFilePath_NUM`, `xQuantTargetMsFilePath_NUM` keys and their value will be the files path. Where NUM is an identification number.
Can be introduced different source or target inputs changing the property NUM suffix.

Example:
```
# Source
xQuantSourcePeptidesFilePath_1 = /xQuant/01/03550.csv
xQuantSourceMsFilePath_1 = /xQuant/01/03550.mzXML
# Target
xQuantTargetPeptidesFilePath_1 = /xQuant/01/03565.csv
xQuantTargetMsFilePath_1 = /xQuant/01/03565.mzXML
xQuantTargetPeptidesFilePath_2 = /xQuant/01/03551.csv
xQuantTargetMsFilePath_2 =/xQuant/01/03551.mzXML
```

The search file must contain the columns: peptide sequence, MS2 scan, peptide charge, peptide id, run id, search id, and theoretical mz.
All searches must have the same columns order.

For a `SQUANT` action, only one input file must be provided. This input file will be the peptide search result file path. This property will be represented by the `inputPeptidesFilePath` and its value will be the file path.

Example:

```
# SQUANT
inputPeptidesFilePath = /executions/squant.csv
```

The search file must contain the columns: peptide sequence, MS2 scan, peptide charge, peptide id, run id, search id, theoretical mz, peptide noise, and a peak column for every label. These peak columns must be adjacent to each other.
As is a single quantification only one row is needed.

The `"PeptidesFile"` csv columns must be indicated using the `input.csv.columns` property. The property must indicate the column position of the peptide sequence, the MS2 scan, the peptide charge, the peptide id, the run id, the search id, and theoretical mz in this order and starting at 0. The single quantification needs two additional columns: noise, start peak (column where can be found the first label peaks) and end peak (column where can be found the last label peaks).

Example:

```
;Quant
# indicated as      Sequence MS2 Charge Id RunId SearchId TheoMz
#input.csv.columns =    12      3   10    5    1      0       9
;sQuant
# indicated as       Sequence MS2 Charge Id RunId SearchId TheoMz Noise sPeak
ePeak
#input.csv.columns =     6       2     5     3     1      0       4      7     8
10
```

- Output

Another needed configuration file property is the output path. The property name is `outputFilePath` and its value will be the path + filename where the quantification algorithm will store the output.

Example:

```
outputFilePath = /Documents/outputs/outputQuant.csv
```

Another output property is the path where the quantification algorithm in `SQUANT` mode will store the output images. The property name is `outputImgPath` and its value is a path, but in this case,

the filename should not be provided.

Example
```
outputImgPath = /Pictures/SQUANT/
```

- Mass modifications
The algorithm has three different modifications: Label modifications, dynamic modifications and static modification.

Label modifications
In order to execute the quantification software, is mandatory to have at least one label defined. To define a label in the configuration file, you should create a new property using the `label_NAME` key. This way, for a label named light, the property key will be `label_Light` and its value will be the amino acid modifications; where each modification will have the amino acid modified, the additional mass and the symbol that identifies the modification. If the label does not have modifications, only a _ will be written as value.

Example:
```
label_Light = _
label_Heavy = K 8.01419881 #
```

In this case, the Light label does not have modifications, and the Heavy label Lysine has an additional mass of 8.01419881, and it has been identified with the # symbol.

If the label has more than one modification, the other modifications must be in the same line.

Example:
```
label_Heavy = K 8.01419881 # C 4.02510698 @
```

In this case, the Heavy label has two modifications: the Lysine, that has an additional mass of 8.01419881 and it has been identified with the # symbol, and the Cysteine that has an additional mass of 4.02510698 identified by the @ symbol.

Dynamic modifications
Amino acids which mass can be found modified. For every possible modified amino acid a property following the next pattern: `mod_AminoSymbol` (mod_ as prefix and the amino acid + symbol as suffix) should be written. Its value will be composed by the additional mass and an expression that indicates if this modification can be over-modified by a label mark.
The expression can be yes, true or 1 for over-modification, or no, false, 0 for non-over-modification.

Example:
```
mod_M* = 15.9949146221 1
```

In this case, the Methionine can be found modified followed by the symbol * and its additional mass is 15.9949146221. If it is the case, and a label has a methionine modification, its additional mass will be summed too.

Example:
`mod_M* = 15.9949146221 0`

In this case, the Methionine can be found modified followed by the symbol * and its additional mass is 15.9949146221. If it is the case, and a label has a methionine modification, its additional mass will not be summed.

Static modifications
Amino acids that will always be found with its mass modified. This modifications will be identified by the `add_AMINO_NAME` property and their value will be the additional mass.

Example:
`add_C_Cysteine = 57.021464`

In this case the Cysteine will always be found with an additional mass of 57.021464.

- Area calculation
There are different properties for the area calculation:

Method
Method to be used in order to calculate the area. The property will be identified by the `area.method` key and its value will be one of the available methods: `TRAPEZOID` or `GAUSSIAN`.

Example:
`area.method = TRAPEZOID`

If the property is not found in the configuration file, `TRAPEZOID` method will be used.

Isotope
Isotopic distribution isotope that will be used in order to quantify the peptides. The property will be identified by the `area.isotope` key and its value will be the isotope number - starting at 0 (monoisotopic).

Example:
`area.isotope = 0`

In this case, the quantification will be done using the monoisotopic.

If the property is not found in the configuration file, the monoisotopic will be used.

Peptide treatment

Previous to the area calculation, the algorithm does some treatments to the peptide peak in time in order to improve the peptide shape and limits. Nowadays, the algorithm has five available methods: `BASIC`, `SMOOTH`, `SMOOTHLIMITS`, `PPMSCORE` and `PPMSCORESMOOTH`. You can enable or disable a peptide treatment, adding a property following the `area.peptideTreatment.METHOD` pattern as a key and yes, true, 1 for enable, or no, false, 0 for disable as its value.

Example:

```
area.peptideTreatment.BASIC = true
area.peptideTreatment.SMOOTH = 0
area.peptideTreatment.SMOOTHLIMITS = 1
area.peptideTreatment.PPMSCORE = false
area.peptideTreatment.PPMSCORESMOOTH = no
```

By default all peptide treatments are disabled. If there aren't peptide treatment properties in the configuration file the `BASIC` method will be used.

Every peptide treatment will result in a different output. This way, the output file name will be the output name defined in `outputFilePath` followed by the method name.

- Preferences

There are different preferences properties:

Search preferences

There are two types of search preferences: Peptide search and peak search.

Only exists one peptide search property, this property is identified by the `peptideSearch.useMSWindow` key and its value will be yes, true, 1 for enable, or false, no, 0 for disable. Use MS window for peptide searching indicates if the peptide will be searched only inside the given scan window or if will be searched without scan limits.

There are three different properties for the peak searching. This properties will be identified by `peakSearch.variableMz`, `peakSearch.similarIntensity`, `peakSearch.similarTheoreticalMz` keys and their value will be yes, true, 1 for enable, or false, no, 0 for disable.

If the variable mz is enabled, the peptide peaks will be searched using a variable mz, this means that if in the actual scan the peptide was found with a 12.8 mz, in the next scan will be searched within the 12.8 +/- ppm window instead of the theoretical mz +/- ppm window. This methodology follows the theory than adjacent scans have similar ppm variations.

If similar intensity is enabled, when more than one peak are found in a scan within the peptide mz window, the one that will be used will be the one that has the most similar intensity to its adjacent scan.

The algorithm will only have into account the similar theoretical mz property if similar intensity is disabled. If it is the case and the property is enabled, the peak selected from the possibilities set

will be the one that has the most similar mz to the theoretical one if variable mz is disabled, or the one with the most similar mz to its adjacent scan otherwise.

Example:

```
peakSearch.variableMz = true
peakSearch.similarIntensity = false
peakSearch.similarTheoreticalMz = true      ; only valid if similar intensity
is not enabled
peptideSearch.useMSWindow = true
```

This are not required properties. If they are not found, the algorithm will use the default values:

```
peakSearch.variableMz = false
peakSearch.similarIntensity = true
peakSearch.similarTheoreticalMz = false
peptideSearch.useMSWindow = false
```

Output preferences

There are two different kind of properties: the output time format and the output columns to be shown.

The output time format is identified by the `timeFormat` key and its value can be `SECONDS`, `MINUTES` or `HOURS`.

Example:

```
timeFormat = MINUTES
```

If the property is not found in the configuration file, the default value will be used: `MINUTES`

The output columns properties are the columns that will be written in the output file. In the configuration file the user can enable or disable the column printing writing the property `output.COLUMN` as key, where `COLUMN` is the column name, and yes, true, 1 (enable), or no, false, 0 (disable) as value.

The output column names are: searchId, runId, peptideId, ms2Scan, noise, s/n, firstScan, lastScan, numScans, score, area, maxIntensity, maxIntensityScan, maxIntensityRetT, areaRatio, areaRatioMin, areaRatioMax, intensityRatio, intensityRatioMin, intensityRatioMax, findingScore, and peaks.

Example:

```
output.peptideId = true
output.findingScore = true;
output.peaks = false;
```

All columns are enabled by default. This way, if an output column property is not found, will use this default value.

- Configuration

Finally, the configuration file has some properties that defines some of the algorithm configurations:

Number of threads

Number of threads that the algorithm will create in order to run the quantification. This property is identified by the `numberOfThreads` key and its value will be the threads number. If the property is not defined, the algorithm will use as many threads as the available processors.

Example:

```
numberOfThreads = 1
```

Mz ppm

Peptide search mz ppm tolerance. For a 10 ppm tolerance, the algorithm will understand +/- 10 ppm. The property can be found in the configuration file with the key `mzPPM`; its value must be a positive number.

Example:

```
mzPPM = 10   ; +/- 10 ppm
```

If the property is not found will be used a tolerance of 5.

Isotopic distribution peaks

Minimum number of isotopic distribution peaks that must be found in order to consider a spectra valid. Will be represented by the key `isotopicDistPeaks`, and its value will be a natural number.

Example:

```
isotopicDistPeaks = 2
```

If the property is not found will be used 2 peaks.

Noise scan window

Scan window to have into account for noise calculation purpose. This window will be +/- the introduced number and are the scans that will be used in order to find the noise peaks. The property will be represented in the configuration file by the key `msNoiseWindow`, and a natural number will be its value.

Example:

```
msNoiseWindow = 20
```

If the property is not found the algorithm will use the default number (20).

Noise mz window

Mz window to have into account for noise calculation purpose. The window will be +/- the introduced number and is the window used in order to find noise peaks. The property will be represented in the configuration file by the key `mzNoiseWindow`, and its value will be a natural number.

Example:
```
mzNoiseWindow = 25
```

If the property is not found the algorithm will use the default number (25).

Peptide scan window
Scan window for peptide mz search. This property will only be used if peptideSearch.useMSWindow is enabled. The window will be +/- the introduced number. The property will be represented by `msPeptideWindow` key, and its value will be a natural number.

Example:
```
msPeptideWindow = 20
```

If the property is not found the algorithm will use the default number (20).

Peptide scans not found
Maximum number of consecutive scans where the peptide mz can be not found. If exceeds this number the peptide search will be ended. The property will be represented by `msPeptideScansNotFound` key, and its value will be a natural number.

Example:
```
msPeptideScansNotFound = 2
```

If the property is not found the algorithm will use the default number (2).

Minimum peaks
Peptide minimum number of peaks in time that must have in order to consider a it valid. The property will be represented by the key `minPeaks`, and its value will be a natural number.

Example:
```
minPeaks = 5
```

If the property is not found the algorithm will use the default number (1).

Maximum smooth
Smooth applications maximum number. The property will be represented by the `maxSmooth` key, and its value will be a natural number. This property will only be used if `area.peptideTreatment` `SMOOTH`, `SMOOTHLIMITS` or `PPMSCORESMOOTH` enabled.

Example

```
maxSmooth = 30
```

If the property is not found the algorithm will use the default number (30).

Cross quantification scan window

Scan window from the theoretical peptide precursor scan in the MS data file where the peptide will be searched. In the configuration file, the property will be identified by the `xQuantMsWindow` key, and its value will be a natural number. The window will be +/- the number. This property will only be used in `XQUANT` action.

Example:

```
xQuantMsWindow = 20
```

If the property is not found the algorithm will use the default number (75).

Execution

You can manually execute the quantification software via console. The only requirement is provide the configuration file path.

To execute the quantification software you should execute the command:

```
> java -jar -Xmx4G /path/to/the/jar/quant.jar /path/to/my/configuration/file/file.ini
```

The -Xmx4G option increases the Java memory heap to 4 GB.

The execution will write some debug messages and the output will be written in the configuration file output path.

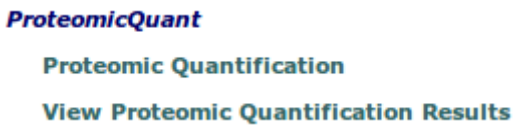
13.2 Module user manual

This document explains to the user how can be executed the proteomics quantification algorithm via the created module, and how the user can view the quantification results.

Quantification Module

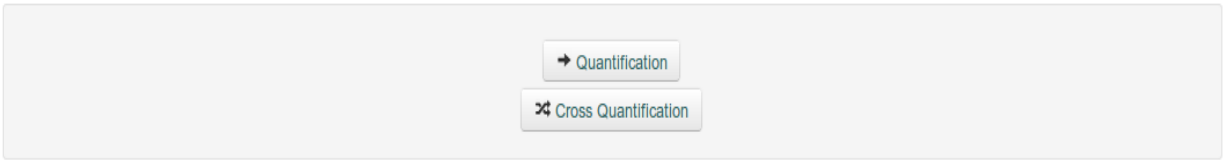
User Manual

The *ProteomicQuant* module is divided in two different sections: the proteomic quantification and the results view.



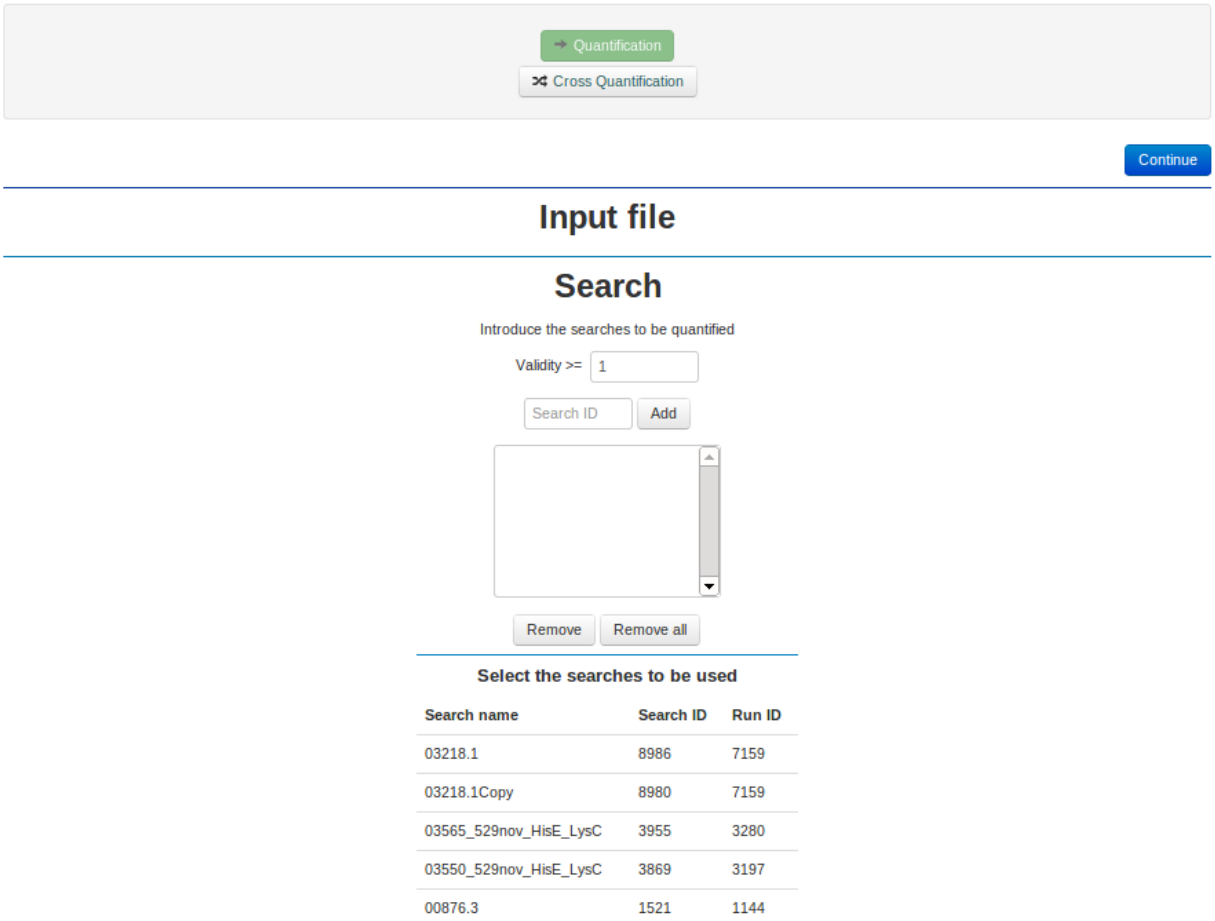
Proteomic Quantification

In this section you can choose between the normal quantification and the cross quantification.



Quantification

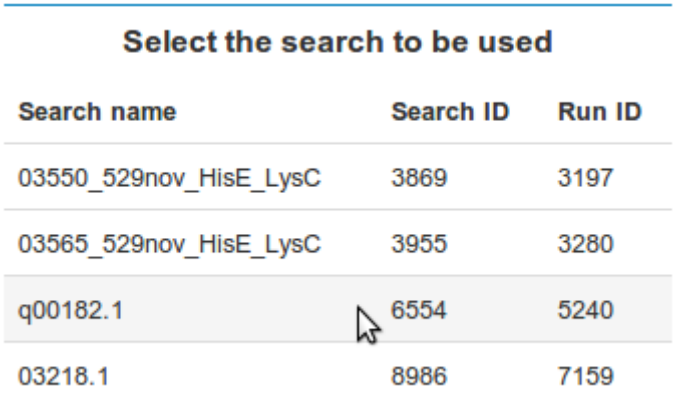
If you have selected the “Normal” Quantification option, a view with all your available searches will appear.



For default will be quantified only the peptides with validity >= 1, if you want to quantify peptides

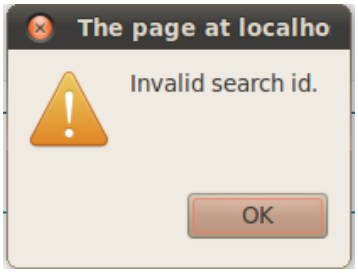
with another validity range, modify the validity input text before selecting the search.

You have to select from the shown table the search from which you want to quantify the peptides.



The other option is directly introduce the search id into the input text and press Add.

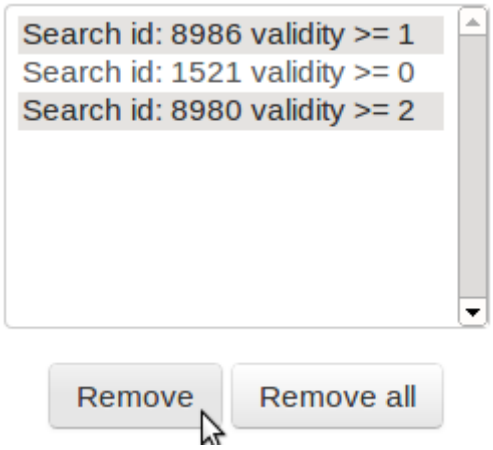
If you don't have enough permissions for the introduced search id, or is not valid, an alert will appear.



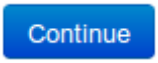
Once you have selected or introduced a valid search, the search will be added into the list.
Every search added, will be seen as a different quantification execution. This is, the program will create a quantification queue.

You can delete all the introduced searches clicking the *Remove all* button.
If you want to delete a concrete search, you have to select the desired one clicking on the name

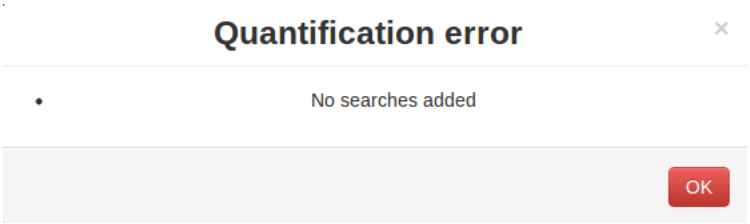
and then clicking the *Remove* button. You can also delete more than one search selecting more than one search with the *Ctrl* or *Command* key pressed.



Once you have all the desired searches in the quantification list, click the *Continue* button located at the top right.



If you haven't selected any search, an error message will be shown.

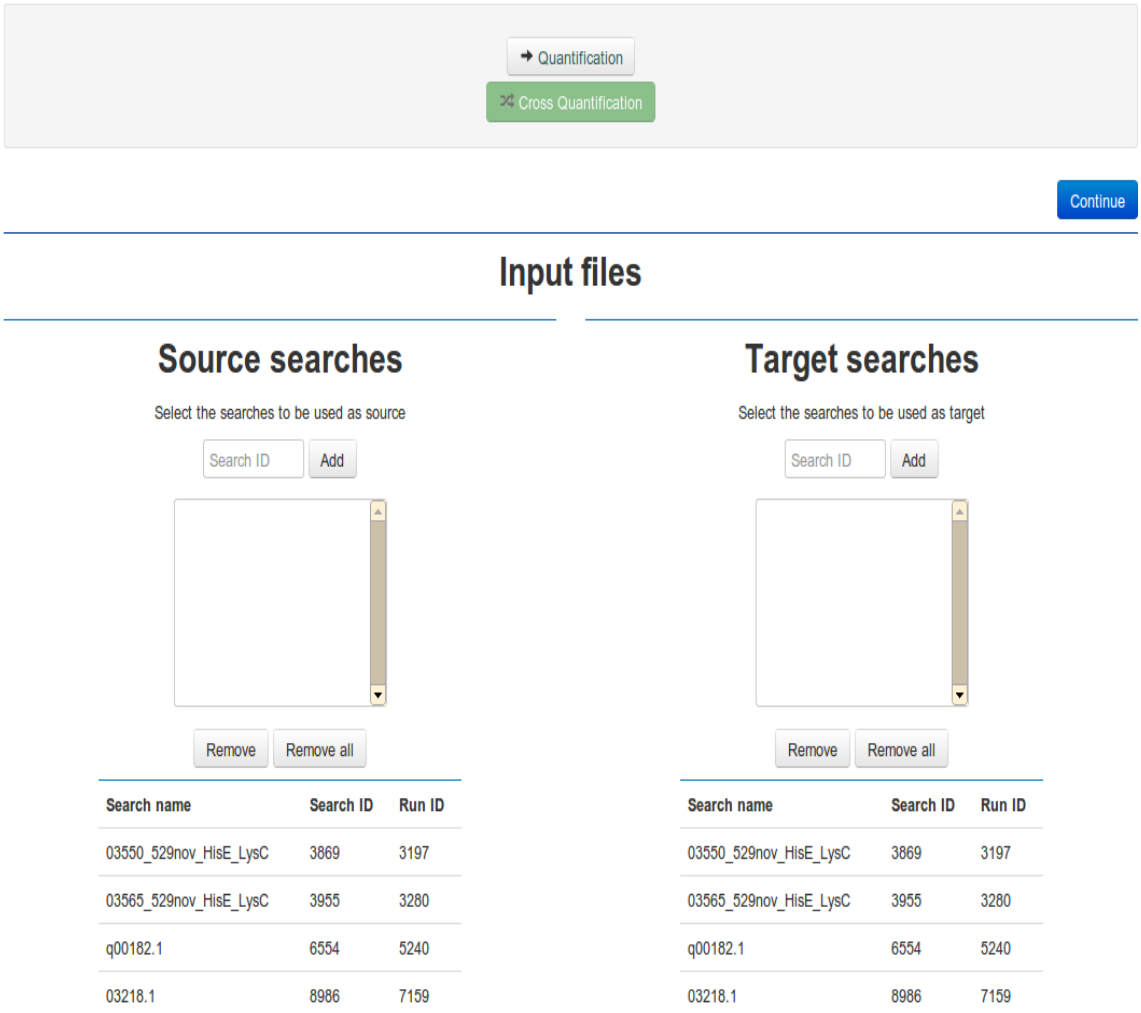


Cross Quantification

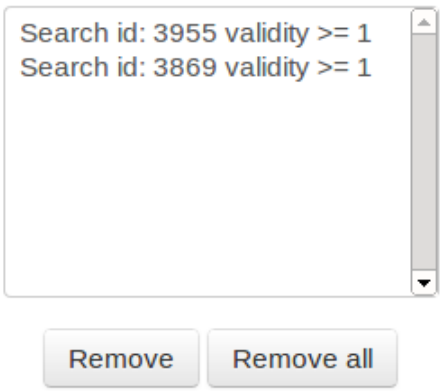
If you have selected the Cross Quantification, a new view will appear. In this window you have to select the source and target searches.

- A source search contains the peptides that you want to quantify.
- A target search is the search in which run you want to find the source peptides.

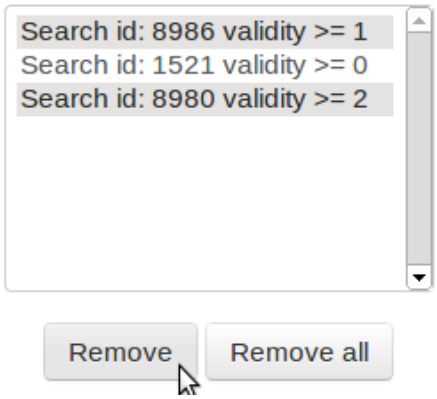
Every target will be seen as a different quantification execution. For this reason, all source peptides will be searched in every target.



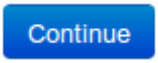
To select a source or a target search, you have to select from its table the desired search, or introduce the search id manually. All the selected searches will appear in the source or target list.



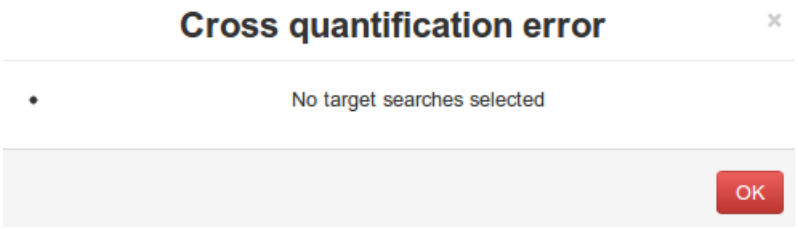
You can delete all the introduced searches clicking the *Remove all* button.
If you want to delete a concrete search, you have to select the desired one clicking on the name and then clicking the *Remove* button. You can also delete more than one search selecting more than one search with the *Ctrl* or *Command* key pressed.



Once you have all source and target searches for the cross quantification, click the *Continue* button located at the top right.

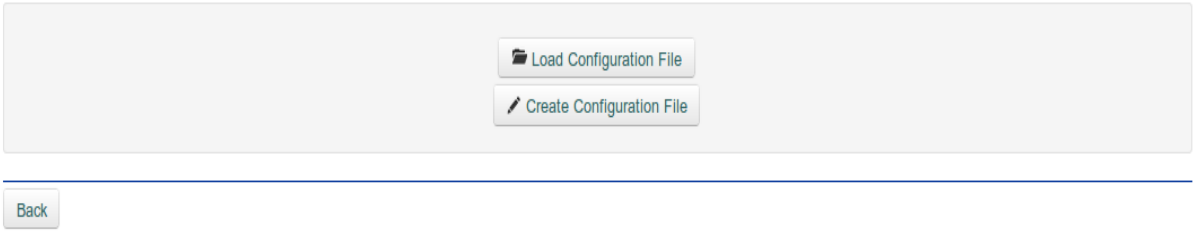


If you haven't selected a source or target search, an error message will be shown.



If everything is ok, the configuration file view will be shown.

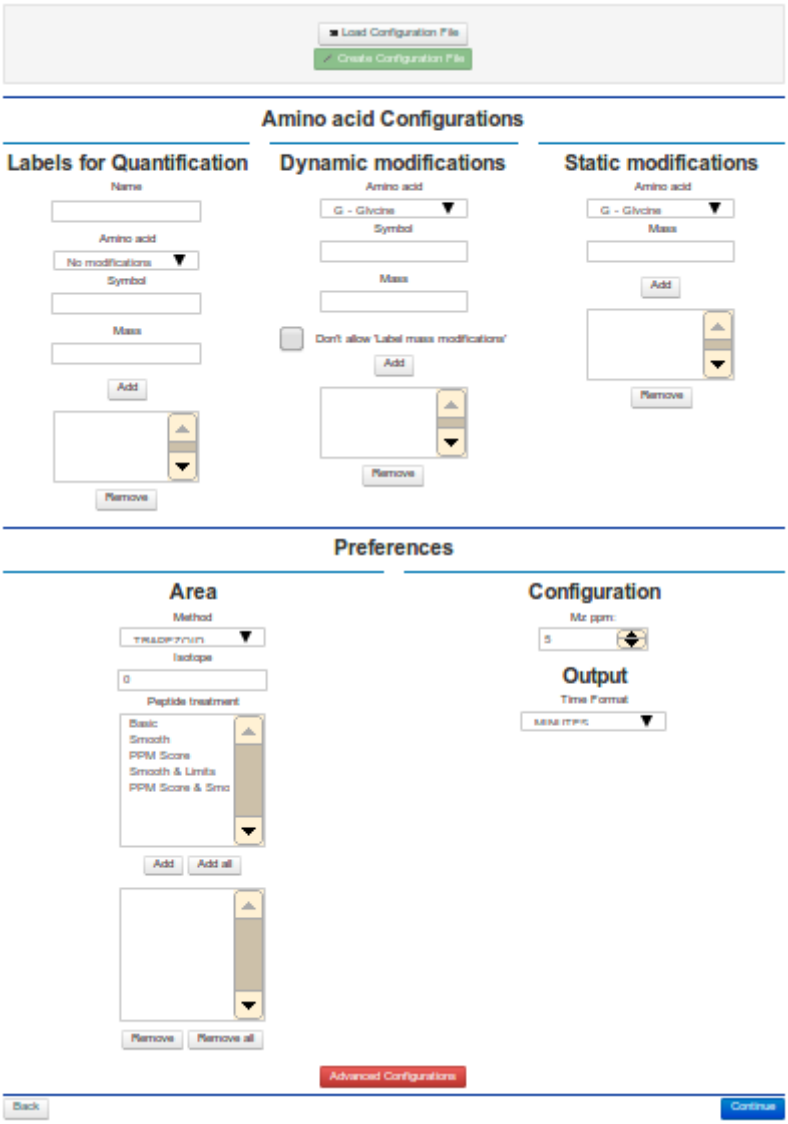
Configuration file
In this section you can chose from an existing file (*Load Configuration File*) or create a new configuration file (*Create Configuration File*).



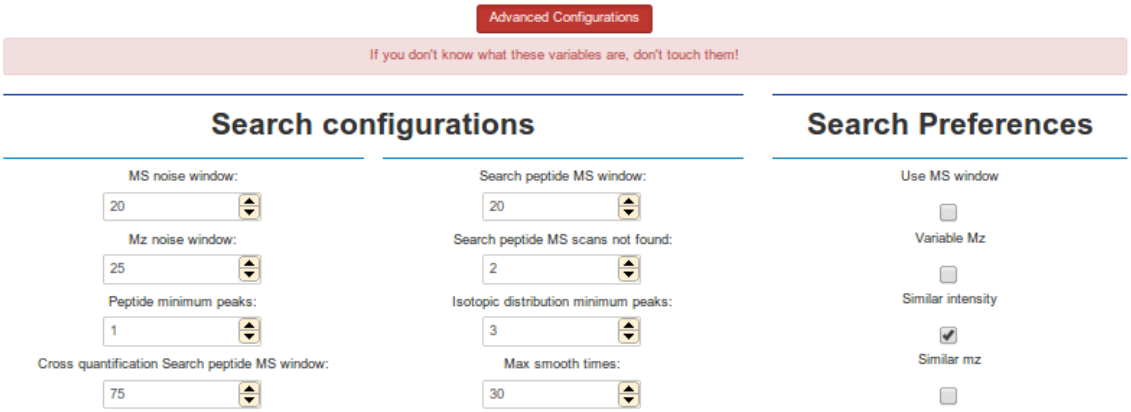
In this section, you can go back to the quantification view at any time clicking the *Back* button located at the bottom left.

Create Configuration file

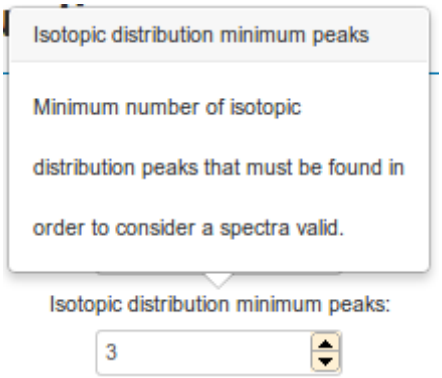
If you have selected the create configuration file option, the following view will be shown:



You can edit the default configuration options pressing the *Advanced Configurations* button.



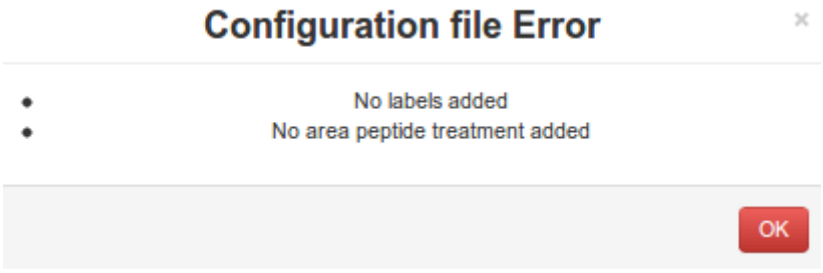
If you click the property name, an explanatory message will appear.



Check the configuration parameters section for more information.

Once you have finished to fill the configuration file, press the *Continue* button.

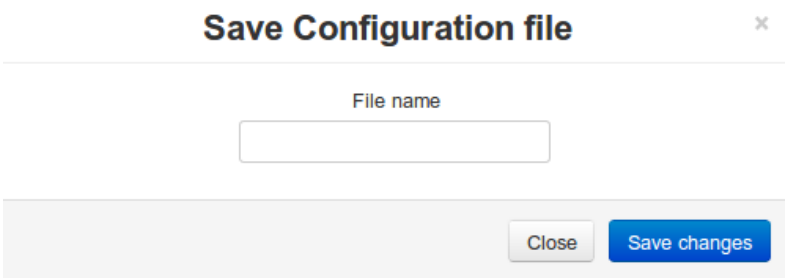
If the introduced data is not valid, a message will be shown.



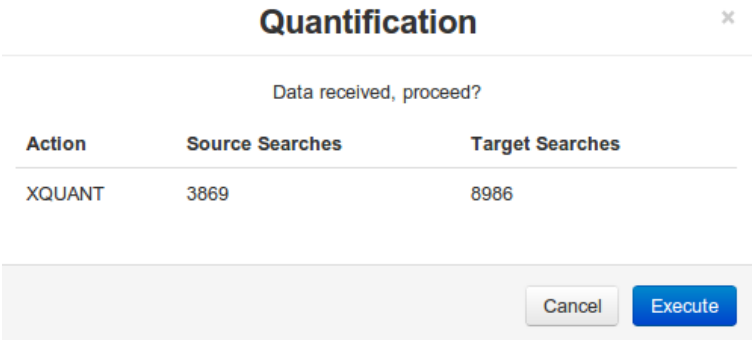
You should have into account that the quantification program needs at least one label and one

method for the area peptide treatment.

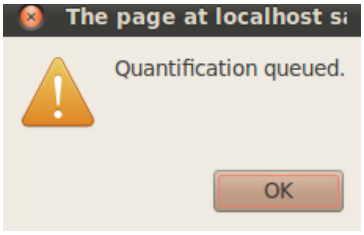
Otherwise, if the introduced data is ok, the following view will be shown:



Introduce the new configuration file name (without blank spaces) and press *Save changes*. You can close this option clicking the *Cancel* button.
After the file saving, a resume table with the quantification action and the searches to be used will be shown.

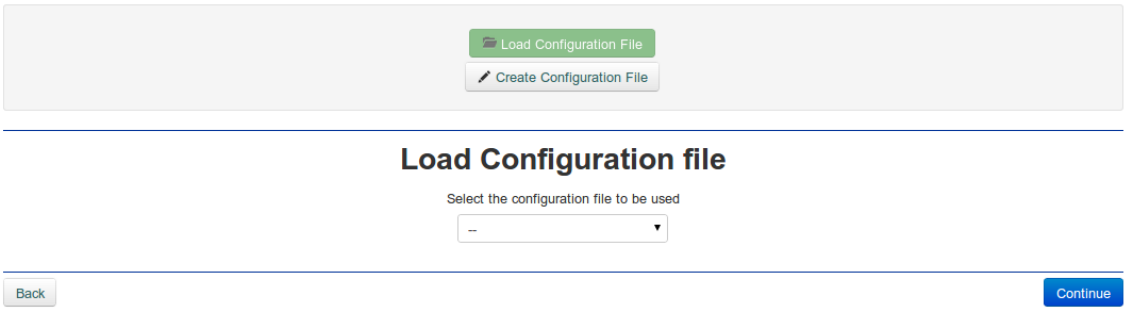


In order to queue the quantification you have to press the *Execute* button. After pressing the button, the cursor will change into wait mode while the module is preparing the data. Finally, an alert will be shown when the quantification is queued.

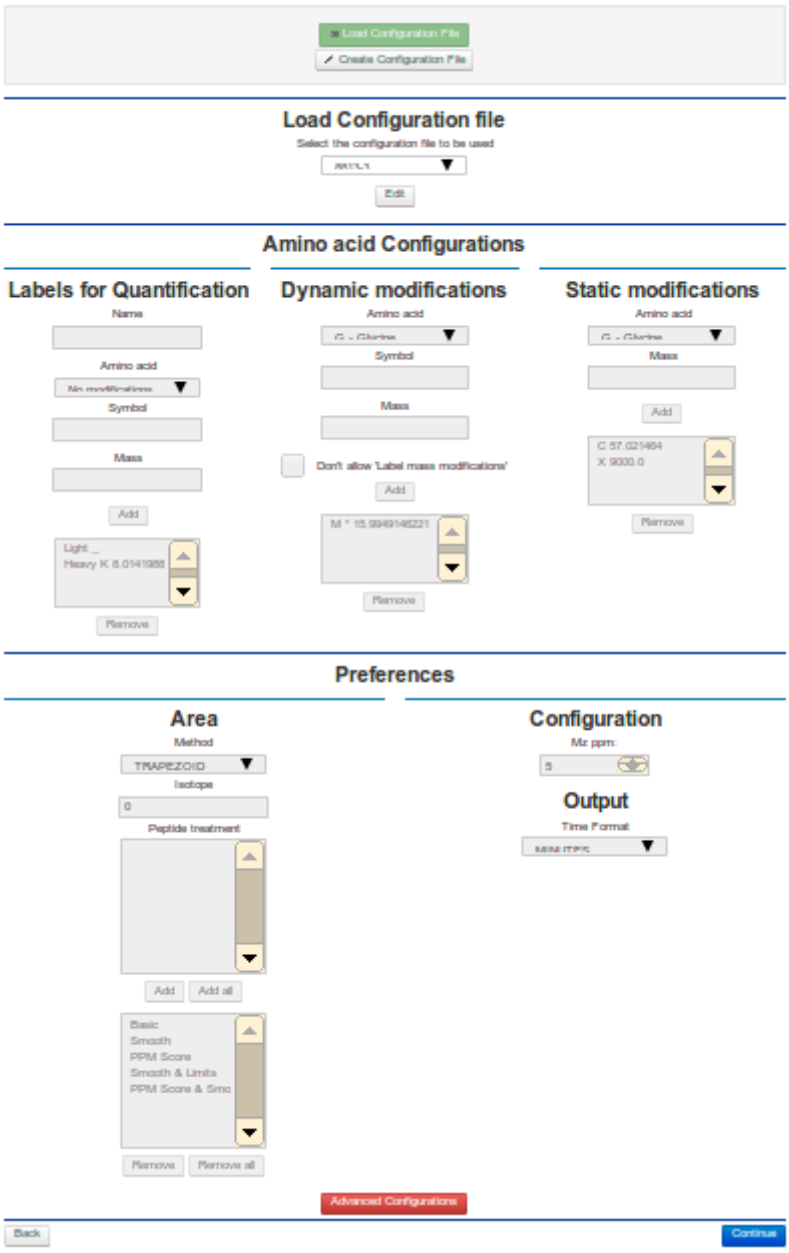


Load Configuration File

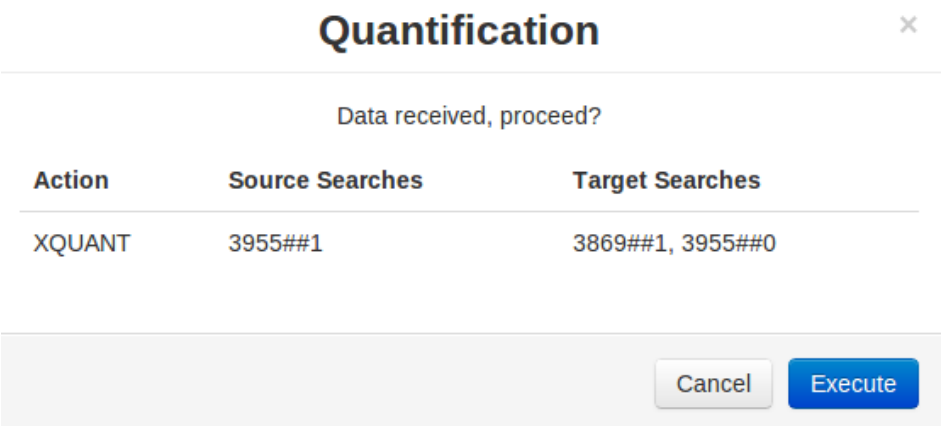
If you have selected the *Load Configuration File* option, will appear a selectable list with all your configuration files stored.



Once you have selected the configuration file, its properties will be shown.

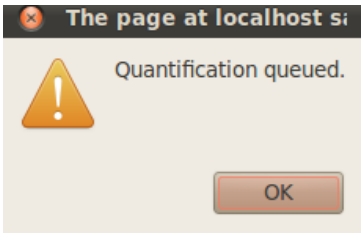


If this configuration parameters are good for you, you can continue by clicking the *Continue* button located at the bottom right.



A resume table with the action and searches will be shown.
The searches are composed by the search id, two pads and the validity used for peptide quantification (search_id##validity).

In order to queue the quantification you have to press the *Execute* button. After pressing the button, the cursor will change into wait mode while the module is preparing the data. Finally, an alert will be shown when the quantification is queued.

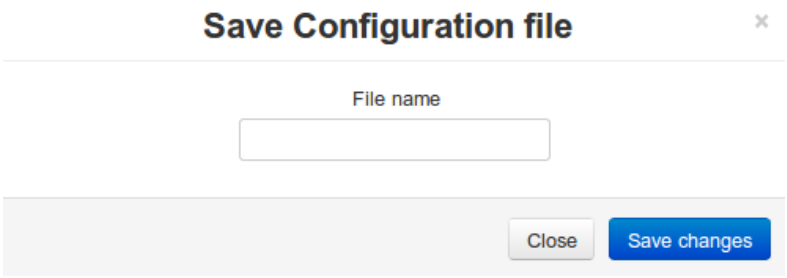


You can cancel the quantification clicking the *Cancel* button, in this case, you will be redirected to the quantification options view.

Loading the configuration file, you can edit the parameters pressing the *Edit* button.

In this case, the configuration file view will be enabled to edit.

You can save the modified configuration file pressing the *Continue* button located at the bottom right. The following view will be shown:

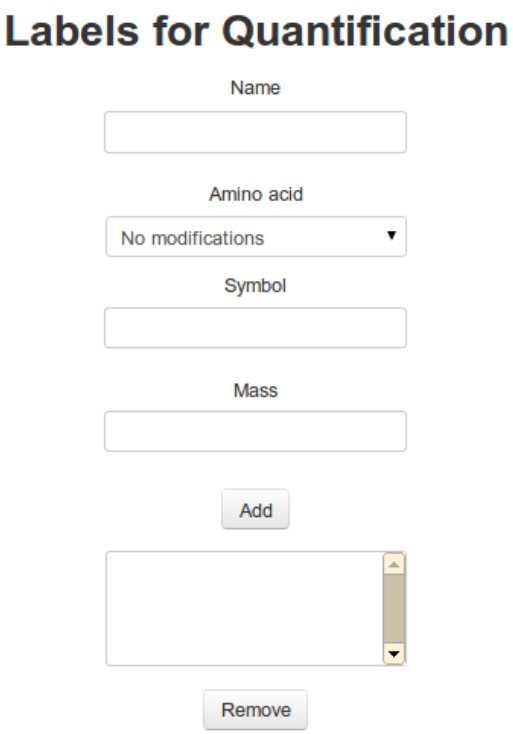


Introduce the new configuration file name (without blank spaces) and press *Save changes*. You can close this option clicking *Cancel*.

After saving the configuration file, the previously explained execution option will be shown.

Configuration parameters

- Labels for Quantification
In this section, you should introduce all labels amino acid modifications.



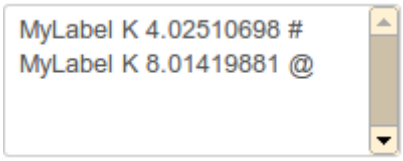
- Name: you can introduce any name (without blank spaces) but always starting by a letter.
- Amino acid: select an amino acid from the list. If your label doesn't have modifications,

select the *No modifications* option.

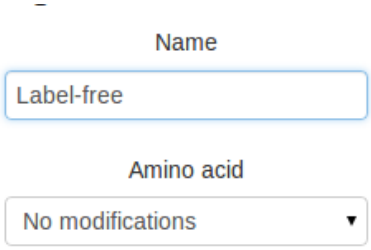
- Symbol: introduce the symbol that identifies the modified amino acid. This symbol cannot be a capital letter.
- Mass: introduce the amino acid additional mass.

Click the *Add* button in order to add the label to the list.

If your label has more than one amino acid modified, you can add it creating a new label with the same name.

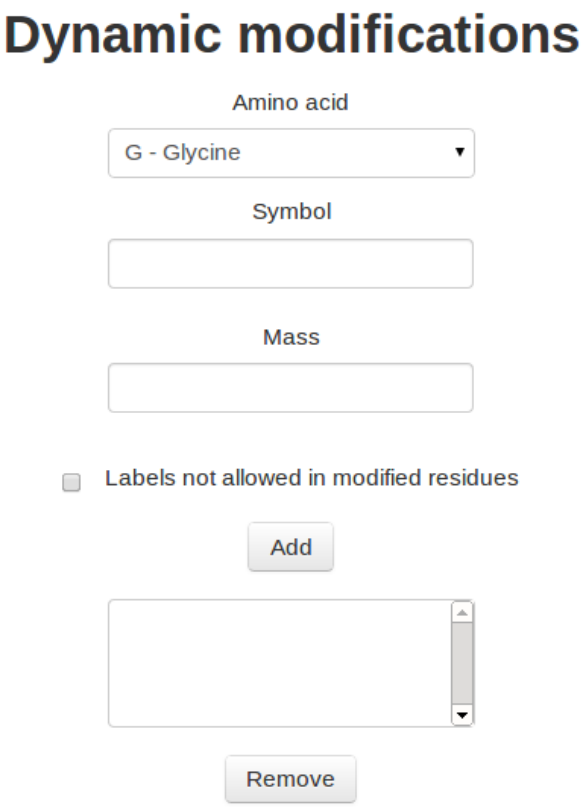


If you want to execute a label-free quantification, you should create a label without modifications.



- Dynamic modifications

In this section you should introduce all your dynamic modifications. This means, that every time that the modification is found, its additional mass will be added.

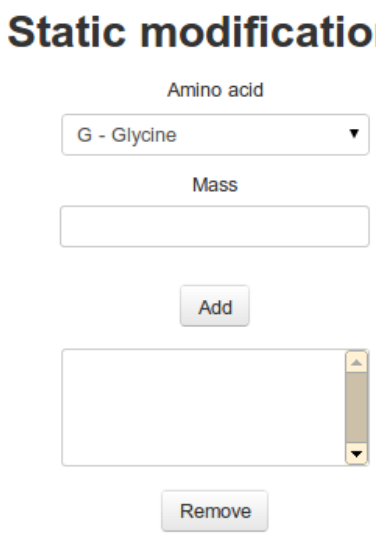


- Amino acid: select the modified amino acid.
- Symbol: select the symbol that identifies the modified amino acid. This symbol can not be a capital letter.
- Mass: the amino acid modification mass.
- Check the '*Labels not allowed in modified residues'* if you have a label with the introduced amino acid marked and you don't want to add its mass when the dynamic modification is found.
  ReDi example:
  Light:](+28)___K*(+42)___K(+28)
  Heavy:](+34)___K*(+42)___K(+34)

The Labels K (+28 or +34) modification is added only when the K* dynamic modification (+42) is not found.

- Static modifications

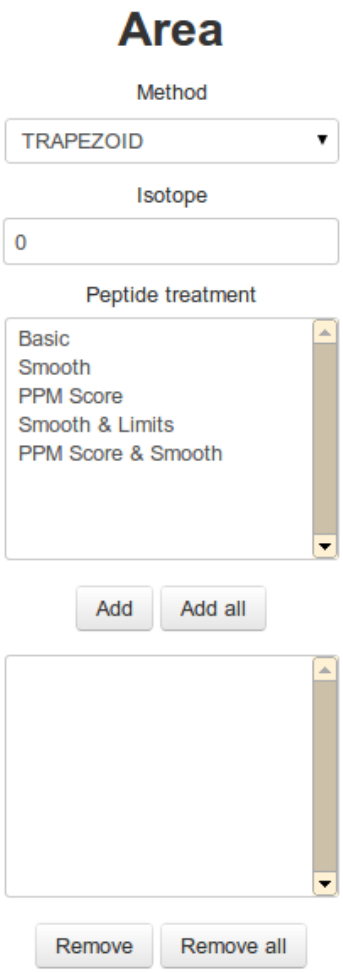
In this section you should introduce all your static modifications. This means that every time that the amino acid is found, its additional mass will be added.



- Amino acid: select the modified amino acid.
- Mass: the amino acid modification mass.

- Area

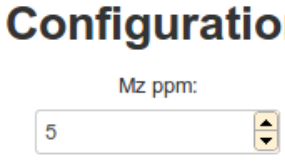
In this section you should introduce your preferences for the area calculation.



- Method: method to be used for the area calculation.
- Isotope: isotope that will be used for the quantification. You should introduce its position, staring by the monoisotopic (0).
- Peptide treatment: Introduce the previous area calculation treatment that you want to do. Every treatment selected will be a different quantification execution.

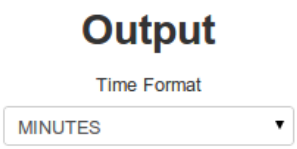
- Configuration

Introduce the peptide mz window ppm. The quantification algorithm will search the peptide within the mz +/- ppm window.



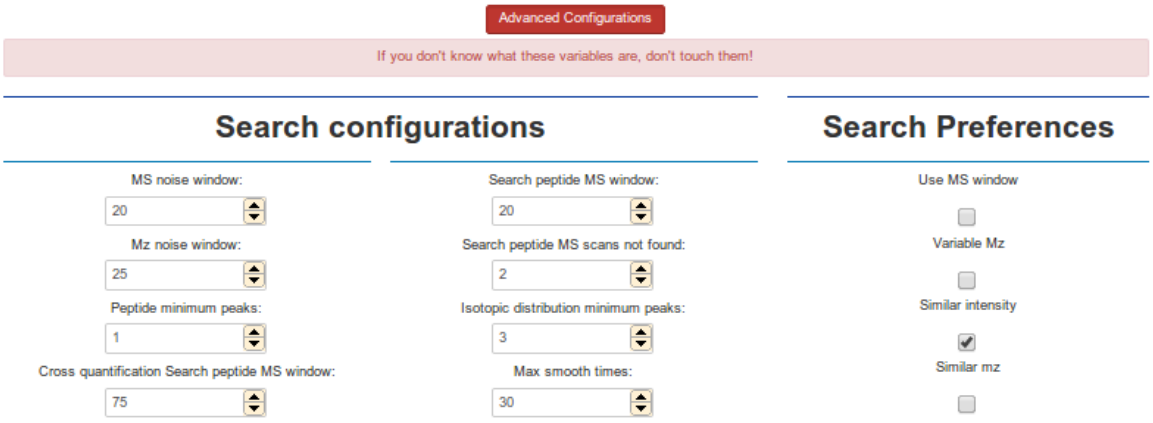
- Output

Introduce the time format: seconds, minutes or hours.



- Advanced Configurations

This properties are directly related to the algorithm execution options. If you don't know what these variables are, do not modify them.



- MS noise window: Scan window that will be used to calculate the noise. This window will be the +/- introduced number.
- Mz noise window: Mz window that will be used in order to calculate the noise. This window will be the +/- introduced number.
- Peptide minimum peaks: Minimum number of peaks that the peptide must have in order to consider it valid.
- Cross quantification Search peptide MS window: Scan window of the MS data file where the peptide mz will be searched when using cross quantification. The window will be +/- the introduced number.
- Search peptide MS window: Scan window for peptide mz search. The window will be +/- the introduced number. This way the maximum number of scans to find the peptide will be 2*introduced number. This property will only be used if use ms window property is enabled.
- Search peptide MS scans not found: Maximum number of consecutive scans where the peptide mz can be not found. If exceeds this number the search will be ended.
- Isotopic distribution minimum peaks: Minimum number of isotopic distribution peaks (additional to the monoisotopic) that must be found in order to consider a spectra valid.
- Max smooth times: Maximum number of smooth applications.

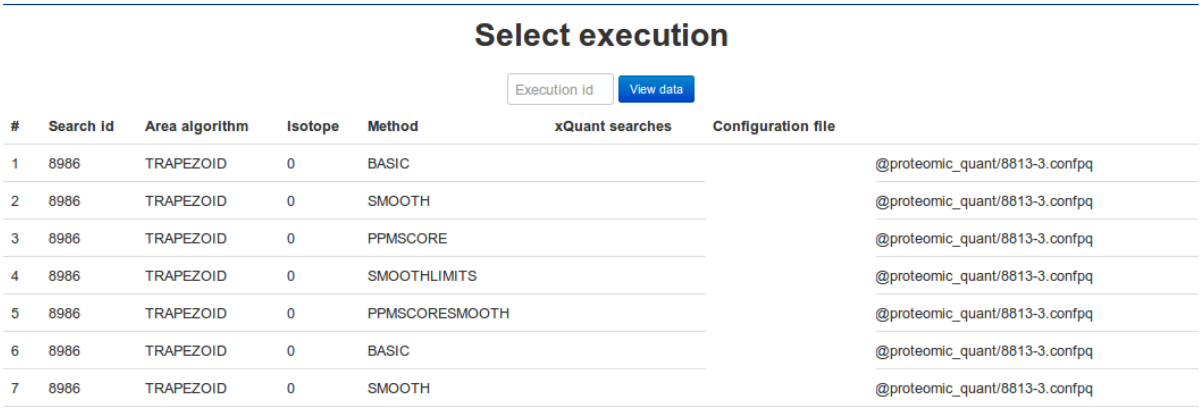
- Use MS window: Using this option, the peptide will only be searched within the introduced MS search peptide window.
- Variable Mz: Uses the theory that the next peak will have a similar ppm to the actual one. This way, will be searched the next peak in the actual mz +/- ppm window.
- Similar intensity: Given two possible peaks for a same scan, the peak with the most similar intensity to its previous will be the selected one.
- Similar mz: Given two possible peaks for the same scan, the peak with the most similar mz to the theoretical mz or its previous peak mz (only if variable mz property is enabled) will be the selected one.

Check the *Algorithm basics* document for further information.

View Proteomic Quantification Results

Executions

In this section you can select the execution from which you want to view the quantification results.



You can also, view the results introducing the execution id into the input text and clicking *View data*.

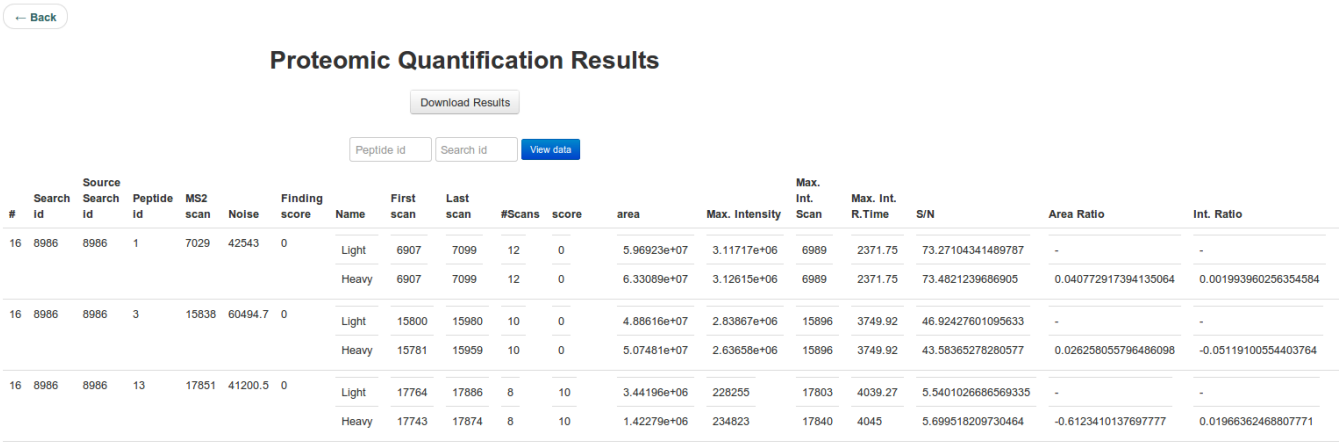
Once we have selected the execution, the proteomic quantification results will be shown.

Proteomic Quantification Results

In this section you can view all peptides quantification results, download them in csv format or check a peptide concrete info.

To download the results, click the *Download Results* button and the download will start.

To view a peptide concrete info click its table row or introduce directly the peptide id and the search id and click the *View data* button.



#	Search id	Source Search Id	Peptide id	MS2 scan	Noise	Finding score	Name	First scan	Last scan	#Scans	score	area	Max. Intensity	Max. Int. Scan	Max. Int. R.Time	S/N	Area Ratio	Int. Ratio
16	8986	8986	1	7029	42543	0	Light	6907	7099	12	0	5.96923e+07	3.11717e+06	6989	2371.75	73.27104341489787	-	-
							Heavy	6907	7099	12	0	6.33089e+07	3.12615e+06	6989	2371.75	73.4821239686905	0.040772917394135064	0.001993960256354584
16	8986	8986	3	15838	60494.7	0	Light	15800	15980	10	0	4.88616e+07	2.83867e+06	15896	3749.92	46.92427601095633	-	-
							Heavy	15781	15959	10	0	5.07481e+07	2.63658e+06	15896	3749.92	43.58365278280577	0.026258055796486098	-0.05119100554403764
16	8986	8986	13	17851	41200.5	0	Light	17764	17886	8	10	3.44196e+06	228255	17803	4039.27	5.5401026686569335	-	-
							Heavy	17743	17874	8	10	1.42279e+06	234823	17840	4045	5.699518209730464	-0.6123410137697777	0.01966362468807771

Peptide Information

In this section you can view the peptide general information, the labels concrete information and a couple of images showing the peptide chromatogram.

- General information: execution id, run id, peptide search id, peptide id, peptide sequence, MS2 scan, peptide charge, peptide mz, peptide noise, algorithm used to calculate the area, isotope used for quantification, and previous area calculation peptide treatment.
- Labels info: every label will have: first scan, last scan, number of scans, score, area, maximum intensity, maximum intensity scan, maximum intensity retention time and its signal to noise value. The “non-light” labels will have the area and intensity ratios too.
- Images: At least will appear two images, the original elution chromatogram and the elution chromatogram obtained after the peptide treatment.

← Back

General information

Execution id: 16

Run id: 7159

Peptide search id: 6986

Peptide id: 1

Sequence: K.EVASLPNK.S

MS2 scan: 7029

Charge: 2

mz: 429.23999

Noise: 42543

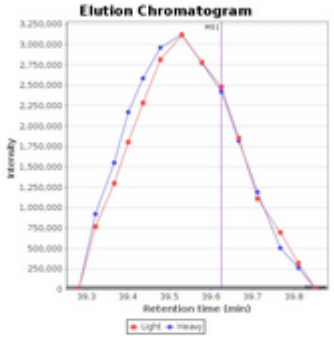
Area algorithm: GAUSSIAN

Isotope used: 0

Peptide Treatment: BASIC

Original peaks

Elution Chromatogram



Labels info

Light

First Scan: 6907

Last Scan: 7099

#Scans: 12

Score: 0

Area: 5.99223e+07

Max. Intensity: 3.11717e+06

Max. Intensity Scan: 6959

Max. Intensity Retention time: 2371.75

S/N: 73.271043414859787

Heavy

First Scan: 6907

Last Scan: 7099

#Scans: 12

Score: 0

Area: 6.33089e+07

Max. Intensity: 3.12615e+06

Max. Intensity Scan: 6959

Max. Intensity Retention time: 2371.75

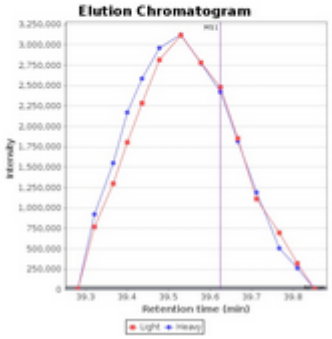
S/N: 73.462123966905

Area Ratio: 0.040772917394135064

Intensity Ratio: 0.0019939602563545B4

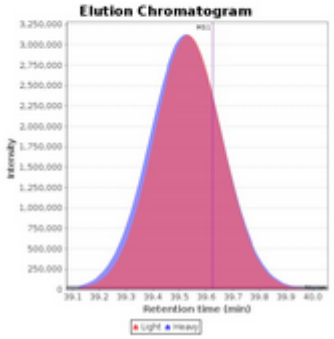
Original peaks

Elution Chromatogram



BASIC

Elution Chromatogram



13.3 Module ajax actions

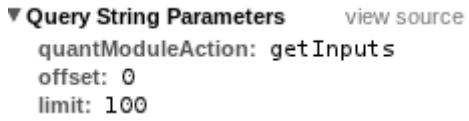
This document shows all the module's existing AJAX actions. This document can be used in order to be able to create a new module, using a different language and maintaining the same client side code.

Quantification Module

Ajax actions

Get inputs
Gets all the user available searches.

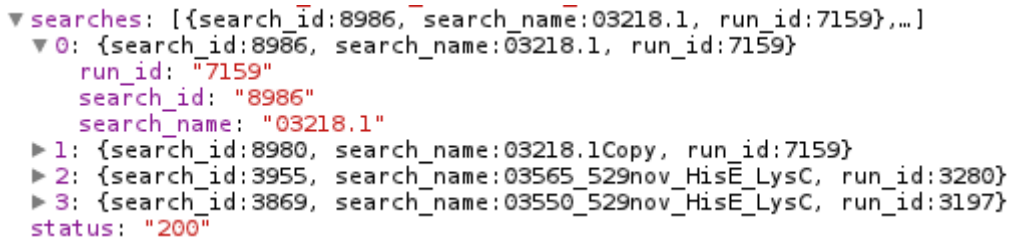
Request
HTTP verb: GET
Data:
- `action`: **`'getInputs'`**
- `offset:` integer that indicates the index of the last search loaded (0 if is the first load)
- `limit:` integer that indicates the maximum number of searches to be loaded.



Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `searches:` array of searches. Each row will contain the run id, search id, and search name indexed by `run_id`, `search_id` and `search_name` respectively.
example:

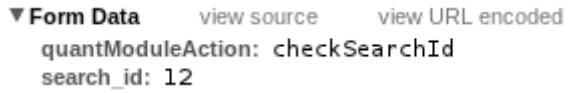
```
searches:
0: {search_id:8986, search_name:03218.1, run_id:7159}
1: {search_id:3955, search_name:03565_529nov_HisE_LysC, run_id:3280}
```

Data Type: json



Check search id
This action checks if the current user has permissions on the given search.

Request
HTTP verb: POST
Data:
- `action`: **`'checkSearchId'`**
- search_id: integer that indicates the search id.

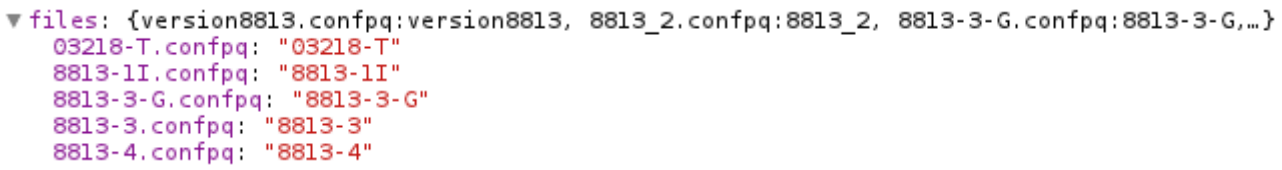


Response
Content:
- `status:` status code (int) that indicates if the user has access to the provided search (200 yes, otherwise no).
- `message:` string with the failure message (only provided if status != 200).
Data Type: json

Load configuration files
Gets all the configuration file names in order to be loaded.

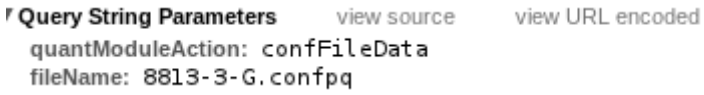
Request
HTTP verb: GET
Data:
- `action`: **`'loadConfigurationFiles'`**

Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- files: array of configuration files. Each row will have the configuration filename (with extension) as key, and the filename (without extension) as value.
example:
`files: {version8813.confpq:version8813, 8813_2.confpq:8813_2,…}`
Data Type: json



Get configuration file data
This action gets the configuration file data in order to be shown.

Request
HTTP verb: GET
Data:
- `action`: **`'confFileData'`**



Response
Content:
- `status`: status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `data`: array containing the quantification file data. Each row will have its own index and data structure. Indexes:
[`addMasses`]: array of dynamic modifications, each row will contain the modified amino acid, the mark symbol, the additional mass, and if it's modifiable or not. These information will be indexed by `amino` (char), `symbol` (char), `mass` (double), `modifiable` (boolean) respectively.
[`areaIsotope`]: integer that indicates the isotope used for the quantification.
[`areaMethod`]: string that indicates the area calculation algorithm.
[`areas`]: array containing the peptide treatments enabled.
[`constMasses`]: array containing the static modifications. Each row will contain the modified amino acid and its additional mass. These information will be indexed by `amino` (char) and `mass` (double) respectively.
[`labels`]: array of labels. Each row will be a different label containing an array of modifications. If the label does not have any modification will be an empty array, otherwise each row will contain the modified amino acid, the mark symbol and its additional mass. These information will be indexed by `amino` (char), `symbol` (char), and `mass` (double) respectively.
[`searchConf`]: array containing the peptide search configurations. Each parameter will be a different row and its value will be an integer (`isotopicDistPeaks`, `maxSmooth`, `minPeaks`, `msNoiseWindow`, `msPeptideScansNotFound`, `msPeptideWindow`, `mzNoiseWindow`, `mzPPM`, and `xQuantMsWindow`).
[`searchPref`]: array containing the peptide search preferences. Each parameter will be a different row and its value will be a boolean depending of if it's enabled or not (`similarIntensity`, `similarTheoreticalMz`, `useMsWindow`, `variableMz`).
[`timeFormat`]: string that indicates the time format (`SECONDS`, `MINUTES`, `HOURS`).
Data Type: json

```
▼ {status:200, data:{labels:{Light:[], Heavy:[{amino:K, mass:8.01419881, symbol:#}]},…}}
  ▼ data: {labels:{Light:[], Heavy:[{amino:K, mass:8.01419881, symbol:#}]},…}
    ▶ addMasses: [{amino:M, symbol:*, mass:15.9949146221, modifiable:true}]
      areaIsotope: "0"
      areaMethod: "GAUSSIAN"
    ▶ areas: [BASIC, SMOOTH, PPMSCORE, SMOOTHLIMITS, PPMSCORESMOOTH]
    ▶ constMasses: [{amino:C, mass:57.021464}, {amino:X, mass:9000.0}]
    ▶ labels: {Light:[], Heavy:[{amino:K, mass:8.01419881, symbol:#}]}
    ▼ searchConf: {mzPPM:5, isotopicDistPeaks:3, msNoiseWindow:20, mzNoiseWindow:25, msPeptideWindow:20,…}
        isotopicDistPeaks: "3"
        maxSmooth: "30"
        minPeaks: "1"
        msNoiseWindow: "20"
        msPeptideScansNotFound: "2"
        msPeptideWindow: "20"
        mzNoiseWindow: "25"
        mzPPM: "5"
        xQuantMsWindow: "75"
    ▼ searchPref: {variableMz:false, similarIntensity:true, similarTheoreticalMz:false, useMSWindow:false}
        similarIntensity: true
        similarTheoreticalMz: false
        useMSWindow: false
        variableMz: false
      timeFormat: "MINUTES"
    status: "200"
```

Use configuration file

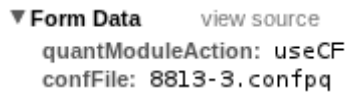
Action used in order to indicate the configuration file that will be used for the quantification algorithm.

Request

HTTP verb: POST

Data:

- `action`: **`'createCF'`**
- `confFile:` configuration file name with extension (ex: `8813-3.confpq`).



Response

Content:

- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).

Data Type: json

Create configuration file
Action used in order to create a new configuration file.

Request
HTTP verb: POST
Data:
- `action`: **`'createCF'`**
- `confFileName`: string containing the configuration file name (ex. `'conf1-2'`)
- `labels`: array of labels. Every row is a string containing the label name, the amino acid modified, its additional mass, and the mark symbol. If the label does not have modifications, the string will be the label name followed by an underscore. Also, if a label has more than one modification, will appear in a different row.
example:

```
labels[]: Light _
labels[]: Heavy K 8.01419881 #
labels[]: Heavy X 2 %
```

- `masses:` array of dynamic modifications. Each row is a string containing the amino acid modified, the symbol used as mark, its additional mass and if the amino acid is modifiable (modifiable) or not (non-modifiable) by a label mark.
example:

```
masses[]: M * 15.9949146221 modifiable
masses[]: Q @ 23.21 non-modifiable
```

- `constants:` array of static modifications. Each row is a string containing the amino acid and its additional mass.
example:

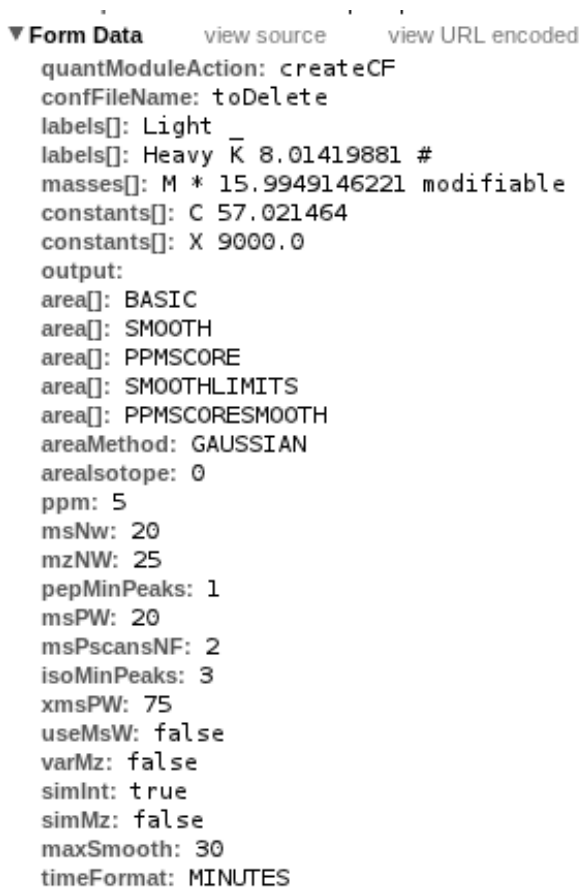
```
constants[]: C 57.021464
constants[]: X 9000.0
```

- area: array of peptide treatments for area calculation. Each row will be a string containing the peptide treatment identifier.
example:

```
area[]: BASIC
area[]: SMOOTH
```

- areaMethod: string containing the area calculation algorithm (`TRAPEZOID` or `GAUSSIAN`).
- `areaIsotope:` integer that indicates the position of the isotope that will be used for area calculation (starting at 0: monoisotopic)
- `ppm:` integer that indicates the ppm tolerance window.
- `msNw:` integer that indicates the noise scan window.
- `mzNW:` integer that indicates the noise mz window.
- `pepMinPeaks:` integer that indicates the minimum number of peaks in order to consider a peptide valid.
- `msPW:` integer that indicates the scan window used for peptide finding.
- `msPscansNF:` integer that indicates the number of consecutive scans where the scan can be not found.
- `isoMinPeaks:` integer that indicates the number of isotopes that the spectra must have

(additionally to the monoisotopic).
- `xmsPW:` integer that indicates the cross quantification scan window for peptide finding.
- `useMsW:` boolean that indicates if the scan window for peptide finding is enabled.
- `varMz:` boolean that indicates if will be used a variable mz for peptide finding.
- `simInt:` boolean that indicates if the intensity will be used to choose between two peaks in the peptide finding.
- `simMz:` boolean that indicates if the mz will be used to choose between two peaks in the peptide finding.
- `maxSmooth:` integer that indicates the maximum number of smooth permitted.
- `timeFormat:` string that indicates the time format (`SECONDS`, `MINUTES` or `HOURS`)



Response

Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `filePath:` path where the configuration file was stored.

Data Type: json

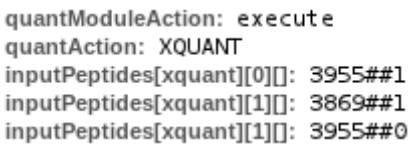
Execute quantification
This action queues the quantification execution.

Request
HTTP verb: POST
Data:
- `action`: **`'execute'`**
- `quantAction:` string that indicates the quantification action (`QUANT, XQUANT`).
- `inputPeptides:` array of searches to be used for quantification. If the action is `QUANT`, the array will be indexed by the `'quant'` key, otherwise the key will be `'xquant'`. For quantification only the searches to quantify will be provided, while in the cross quantification the target and sources will be provided in different rows (sources: row 0, targets: row 1).
example QUANT:

```
inputPeptides[quant][]: 8980##1
inputPeptides[quant][]: 8986##1
```

example XQUANT:

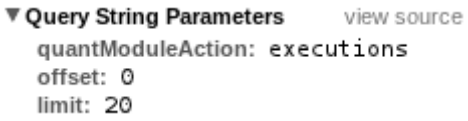
```
inputPeptides[xquant][0][]: 3955##1
inputPeptides[xquant][1][]: 3869##1
inputPeptides[xquant][1][]: 3955##0
```



Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
Data Type: json

Get executions
Gets the user quantification executions.

Request
HTTP verb: GET
Data:
- `action`: **`'executions'`**
- `offset:` integer that indicates the index of the last execution loaded (0 if is the first load)
- `limit:` integer that indicates the maximum number of executions to be loaded.

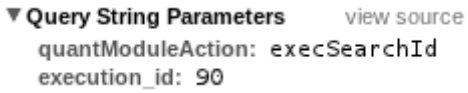


Response
Content:
- `executions:` array containing the executions info. Each row will be a different execution and will contain an array with the area calculation algorithm (`area_algorithm`), the isotope used for quantification (`area_isotope`), the used configuration file path (`config_path`), the execution id (`exec_id`), the peptide treatment used (`method`), the search id (`search_id`), and the cross quantification used searches (`xquant_searches`).
Data Type: json

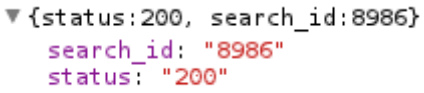

```
▼ executions: [{exec_id:102, search_id:3869, method:PPMSCORESMOOTH, area_algorithm:GAUSSIAN, area_isotope:0,…},…]
  ▼ 0: {exec_id:102, search_id:3869, method:PPMSCORESMOOTH, area_algorithm:GAUSSIAN, area_isotope:0,…}
      area_algorithm: "GAUSSIAN"
      area_isotope: "0"
      config_path: "/lab/user_data/gfyadmini/@proteomic_quant/8813-3-G.confpq"
      exec_id: "102"
      method: "PPMSCORESMOOTH"
      search_id: "3869"
      xquant_searches: "3955"
  ▶ 1: {exec_id:101, search_id:3869, method:SMOOTHLIMITS, area_algorithm:GAUSSIAN, area_isotope:0,…}
  ▶ 2: {exec_id:100, search_id:3869, method:PPMSCORE, area_algorithm:GAUSSIAN, area_isotope:0,…}
```

Get execution search id
Gets the execution search id

Request
HTTP verb: GET
Data:
- `action`: **`'execSearchId'`**
- `execution_id:` integer that identifies the execution id.

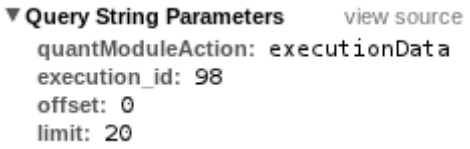


Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `message:` String containing the error message (only if status != 200).
- `search_id:` integer that indicates the execution search id.
Data Type: json



Get execution data
Gets the quantification execution results data.

Request
HTTP verb: GET
Data:
- `action`: **`'executionData'`**
- `execution_id:` integer that identifies the execution id.
- `offset:` integer that indicates the index of the last execution result loaded (0 if is the first load)
- `limit:` integer that indicates the maximum number of execution results to be loaded.

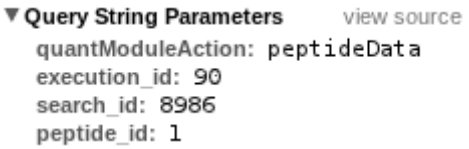


Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `message:` String containing the error message (only if status != 200).
- `data:` array containing the execution results. Each row will contain an execution result; this result will be an array indexed by `info` and `labels`, where `info` will contain the peptide general information, and `labels` an array of labels.
The info array will contain: `findingScore`, `ms2Scan`, `noise`, `peptideId`, and `source_search_id`.
Every row of labels will contain: `area`, `firstScan`, `labelName`, `lastScan`, `maxIntensity`, `maxIntensityRetT`, `maxIntensityScan`, `numScans`, and `score`.
Data Type: json


```
▼ {status:200,…}
  ▼ data: [{info:{peptideId:1, source_search_id:3869, ms2Scan:28932, noise:7753.52, findingScore:0},…},…]
    ▼ 0: {info:{peptideId:1, source_search_id:3869, ms2Scan:28932, noise:7753.52, findingScore:0},…}
      ▼ info: {peptideId:1, source_search_id:3869, ms2Scan:28932, noise:7753.52, findingScore:0}
          findingScore: "0"
          ms2Scan: "28932"
          noise: "7753.52"
          peptideId: "1"
          source_search_id: "3869"
      ▼ labels: [{labelName:Light, firstScan:28822, lastScan:28948, numScans:7, score:0, area:1.44951e+06,…},…]
        ▼ 0: {labelName:Light, firstScan:28822, lastScan:28948, numScans:7, score:0, area:1.44951e+06,…}
            area: "1.44951e+06"
            firstScan: "28822"
            labelName: "Light"
            lastScan: "28948"
            maxIntensity: "95323.5"
            maxIntensityRetT: "5289.64"
            maxIntensityScan: "28906"
            numScans: "7"
            score: "0"
        ▶ 1: {labelName:Heavy, firstScan:28822, lastScan:28969, numScans:8, score:0, area:1.88676e+06,…}
    ▶ 1: {info:{peptideId:5, source_search_id:3869, ms2Scan:32328, noise:10332.4, findingScore:100},…}
    ▶ 2: {info:{peptideId:9, source_search_id:3869, ms2Scan:13724, noise:70150, findingScore:0},…}
```

Get peptide data
This action gets all the peptide information.

Request
HTTP verb: GET
Data:
- `action`: **`'peptideData'`**
- `execution_id:` integer that identifies the execution.
- `search_id:` integer that identifies the peptide search.
- `peptide_id:` integer that identifies the peptide.

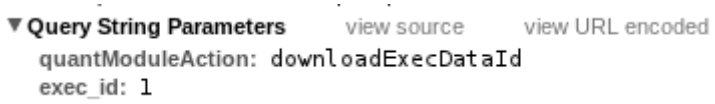


Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `message:` String containing the error message (only if status != 200).
- data: array that contains the quantified peptide information. This array will be indexed by img, info, and labels.
[`img`]: array of images. Each row will be an encoded base64 bitmap image.
[`info`]: array containing the peptide general information. Array keys: `area_algorithm`, `area_isotope`, `findingScore`, `method`, `ms2Scan`, `noise`, `peptide_m_z`, `peptide_sequence`, `peptide_solution_charge`, `run_id`.
[`labels`]: array of labels information. Each row will contain the label `area`, `firstScan`, `labelName`, `lastScan`, `maxIntensity`, `maxIntensityRetT`, `maxIntensityScan`, `numScans`, and `score`.
Data Type: json

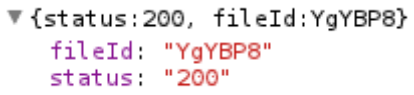
```
▼ data: {info:{ms2Scan:7029, noise:42543, findingScore:0, run_id:7159, peptide_m_z:429.23999,…},…}
  ▶ img: [,…]
  ▼ info: {ms2Scan:7029, noise:42543, findingScore:0, run_id:7159, peptide_m_z:429.23999,…}
      area_algorithm: "GAUSSIAN"
      area_isotope: "0"
      findingScore: "0"
      method: "PPMSCORE"
      ms2Scan: "7029"
      noise: "42543"
      peptide_m_z: "429.23999"
      peptide_sequence: "K.EVASLPNK.S"
      peptide_solution_charge: "2"
      run_id: "7159"
  ▼ labels: [{labelName:Light, firstScan:6907, lastScan:7099, numScans:12, score:0, area:5.96923e+07,…},…]
    ▼ 0: {labelName:Light, firstScan:6907, lastScan:7099, numScans:12, score:0, area:5.96923e+07,…}
        area: "5.96923e+07"
        firstScan: "6907"
        labelName: "Light"
        lastScan: "7099"
        maxIntensity: "3.11717e+06"
        maxIntensityRetT: "2371.75"
        maxIntensityScan: "6989"
        numScans: "12"
        score: "0"
    ▶ 1: {labelName:Heavy, firstScan:6907, lastScan:7099, numScans:12, score:0, area:6.33089e+07,…}
  status: "200"
```

Get execution download file id
Gets the quantification results file identifier in order to be downloaded.

Request
HTTP verb: GET
Data:
- `action`: **`'downloadExecDataId'`**
- `exec_id:` integer that identifies the execution from which the results will be downloaded.



Response
Content:
- `status:` status code (int) that indicates if the action was successful or not (200 ok, otherwise failed).
- `fileId:` string that identifies the file to be downloaded.
Data Type: json



13.4 Configuration file

The algorithm has a set of configuration and preference variables that can be set by the user in order to accurate the algorithm for his needs.

The configuration file format is a mix of java property files and php ini files.

Format
Reads a property list (key and element pairs) from the given input path. Properties are processed in terms of lines. A line is defined as a line of characters that is terminated either by a set of line terminator characters (\n, \r or \r\n) or by the end of the stream. A natural line may be either a blank line or a comment line.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#', '[' or ';' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab ('\t'), and form feed ('\f') to be white space.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', or white space character other than a line terminator. Any white space after the key is skipped; if the first non-white space character after the key is '', then it is ignored and any white space characters after it are also skipped. All remaining characters found before a ';' on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". A ';' character is seen as a comment indicator; all text written in the line after this character will be ignored.

Lets see a correct configuration file example:

```
[SINGLE QUANTIFICATION]
inputPeptidesFilePath = /path/file.ext
inputPeptideFileFormat=/path/file.ext

[MS]
ppm =10 ; ppm tolerance: 10 is +/- 10
```

```
#msWindow
 minPeaks= 5
             !maxSmooth = 4
```

From this configuration file, will be obtained:

```
inputPeptidesFilePath → /path/file.ext
inputPeptideFileFormat → /path/file.ext
ppm → 10
minPeaks → 5
```

All white lines, white spaces and comments were successfully skipped.

Configuration variables
In this section will be shown the different variables that will be used in order to configure properly the quantification algorithm.

- Configuration.constants: amino acid static mass modifications. To successfully identify these static modifications in the configuration file, every amino acid additional mass should be written using the following pattern: `add_ID_AMINOACID`, where `ID` is the amino acid letter identifier (uppercase), and `AMINOACID` the name.
  The regular expression that identifies a constant mass property is: `add_([A-Z][A-Z]?)_([a-zA-Z]+)`

  ```
  Example:
  add_T_Threonine = 0.0000
  add_C_Cysteine = 57.021464
  add_L_Leucine = 0.0000
  add_I_Isoleucine = 0.0000
  add_X_LorI = 9000.0
  ```

  If an amino acid additional mass pattern is not found, will use 0 as its default additional mass.

- Configuration.labels and Configuration.labelNames: This configuration variable represents the additional amino acid masses to be added for every label. For every different label, a key property following the pattern `label_NAME` will be found. These properties will have as a value: the amino acid letter identifier (uppercase), the additional mass, and the mark symbol. If the label does not have modifications a '_' must be added.
  The regular expression that identifies the labels is: `label_([a-zA-Z]+)`
  The values can be defined as:

  ```
  _ | (([A-Z][A-Z]?)( |\t|\f)+([0-9]+(.[0-9]+)?)( |\t|\f)+([^A-Z])
  (( |\t|\f)+([A-Z][A-Z]?)( |\t|\f)+([0-9]+(.[0-9]+)?)
  ( |\t|\f)+([^A-Z]))*)
  ```

  Note that the symbol cannot be an uppercase letter (reserved for amino acids).

  ```
  Example:
  label_Light = _ ; no marks
  label_Medium = K 4.02510698 #
  label_Heavy = K 8.01419881 @
  ```

- Configuration.modifiers: Amino acids which mass can be found modified (dynamic modifications). For every possible modified amino acid will be found a property following the next pattern: `mod_IDSymbol` where `ID` is the amino acid identifier, and `Symbol` the modification mark used. Its value will be the additional mass and an expression that indicates if this modification can be over-modified with a label mark.
  The regular expression that identifies the property name is:
  `mod_([A-Z][A-Z]?)([Â-Z])`
  The value can be expressed as:

  ```
  (.[0-9]+)?)( |\t|\f)+([^A-Z])
  ((([yY]([eE][sS])?)|([tT]([rR][uU][eE])?)|1)
  |(([nN]([oO])?)|([fF]([aA][lL][sS][eE])?)|0))
  ```

```
Example:
mod_M* = 15.9949146221 1
```

- Configuration.ppm: Peptide search mz ppm tolerance. For a 10 ppm tolerance, the algorithm understands an $mz \pm 10$ ppm window.

  ```
  Example:
  mzPPM = 10 ; +/- 10 ppm
  ```

- Configuration.isotopicDistPeaks: Minimum number of isotopic distribution peaks that must be found in order to consider a spectra valid. Will be represented by the key `isotopicDistPeaks`, and its value will be a natural number.

  ```
  Example:
  isotopicDistPeaks = 2
  ```

- Configuration.msNoiseWindow: Scan window to have into account for noise calculation purpose. This window will be +/- the introduced number. The property will be represented in the configuration file by the key `msNoiseWindow`, and a natural number will be its value.

  ```
  Example:
  msNoiseWindow = 20
  ```

- Configuration.mzNoiseWindow: Mz window to have into account for noise calculation purpose. The window will be +/- the introduced number. The property will be represented in the configuration file by the key `mzNoiseWindow`, and its value will be a natural number.

  ```
  Example:
  mzNoiseWindow = 25
  ```

- Configuration.msPeptideWindow: Scan window for the peptide's mz search. The window will be +/- the introduced number and will be the window where the peptide will be searched within. The property will be represented by `msPeptideWindow` key, and its value will be a natural number.

```
Example:
msPeptideWindow = 20
```

- Configuration.msPeptideScansNotFound: Maximum number of consecutive scans where the peptide mz can be not found. If exceeds this number the search will be ended. The property will be represented by msPeptideScansNotFound key, and its value will be a natural number.

  ```
  Example:
  msPeptideScansNotFound = 2
  ```

- Configuration.minPeaks: Minimum number of peaks that the peptide must have in order to consider it valid. The property will be represented by the key `minPeaks`, and its value will be a natural number.

  ```
  Example:
  minPeaks = 5
  ```

- Configuration.maxSmooth: maximum number of smooth applications. The property will be represented by the `maxSmooth` key, and its value will be a natural number.

  ```
  Example
  maxSmooth = 30
  ```

- Configuration.numberOfThreads: Number of threads to be used in the quantification algorithm. The property will be represented by the `numberOfThreads` key and its value will be a natural number.

  ```
  Example:
  numberOfThreads = 16
  ```

- Configuration.inputPeptidesFilePath: Peptides data file path (quantification input). The property will be represented by the `inputPeptidesFilePath`, and its value will be the file path.

  ```
  Example:
  inputPeptidesFilePath = /home/ari/Documents/csv/03218.csv
  ```

- Configuration.inputMsFilePath: Mass spectrometer's data file path (quantification input). The property will be represented by the `inputMsFilePath`, and the file path will be its value. This file must be the one from which the input peptides were extracted.

  ```
  Example:
  inputMsFilePath = /home/ari/Documents/mzxml/03218.mzXML
  ```

- Configuration.xSourcePeptidesFilesPath: Peptides data files path; Source runs peptides for cross quantification. In the configuration file, this property will be represented by the key pattern `xQuantSourcePeptidesFilePath_NUMBER`, where `NUMBER` is a natural number. Its value will be the file path.
  The regular expression that identifies the property is:
  `xQuantSourcePeptidesFilePath_[0-9]+`
  Important: All files must have the same format.

  ```
  Example:
  xQuantSourcePeptidesFilePath_1 = /home/ari/Documents/csv/0222.csv
  xQuantSourcePeptidesFilePath_2 = /home/ari/Documents/csv/0522.csv
  xQuantSourcePeptidesFilePath_3 = /home/ari/Documents/csv/0224.csv
  ```

- Configuration.xSourceMsFilesPath: Mass spectrometry data file path; source MS data for cross quantification. In the configuration file, the property will be represented by the pattern `xQuantSourceMsFilePath_NUMBER`, where `NUMBER` is a natural number and must be the same as their peptides source files.
  The regular expression that identifies the property is: `xQuantSourceMsFilePath_[0-9]+`
  Important: All files must have the same format.

  ```
  Example:
  xQuantSourceMsFilePath_1 = /home/ari/Documents/mzxml/0222.mzXML
  xQuantSourceMsFilePath_2 = /home/ari/Documents/mzxml/0522.mzXML
  xQuantSourceMsFilePath_3 = /home/ari/Documents/mzxml/0224.mzXML
  ```

- Configuration.xTargetPeptidesFilesPath: Peptides data files path; Peptides of the corresponding MS target runs for cross quantification. The configuration file key will be `xQuantTargetPeptidesFilePath_NUMBER`, where `NUMBER` is a natural number, and its value is the file path.

The regular expression that identifies the property is:
`xQuantTargetPeptidesFilePath_[0-9]+`
All files must have the same format, and must be the same format as `xSourcePeptidesFilesPath`.

Example:
`xQuantTargetPeptidesFilePath_1 = /home/ari/Documents/csv/03442.csv`

- Configuration.xTargetMsFilesPath: Mass spectrometry data file path; target MS data for cross quantification. In the configuration file, the property will be represented by the pattern `xQuantTargetMsFilePath_NUMBER`, where `NUMBER` is a natural number, and must be the same as the peptides list input that uses this MS file.
  The regular expression that identifies the property is: `xQuantTargetMsFilePath_[0-9]+`
  All files must have the same format, and must be the same as xSourcePeptidesFilesPath.

  Example:
  `xQuantTargetMsFilePath_1 = /home/ari/Documents/mzxml/03442.mzXML`

  Note that the keys that have the same number, are from the same run.

- Configuration.xMsWindow: Scan window in order to search the peptide mz in the MS data file when the cross quantification action is performed. In the configuration file, the property will be identified by the `xQuantMsWindow` key, and its value will be a natural number. The window will be +/- the number.

  Example:
  `xQuantMsWindow = 20`

- Configuration.outputFilePath: Quantification output file path. In the configuration file will be represented by the `outputFilePath` key, and its value will be the file path.

  Example:
  `outputFilePath = /home/ari/Documents/output/03218.csv`

- Configuration.outputImagePath: Quantification output image path. In the configuration file will be represented by the `outputImgPath` key, and its value will be the image path without the file name.

  ```
  Example:
  outputImgPath = /home/ari/Pictures/prova/
  ```

- Configuration.inputColumns: This property is used only if the peptides list input is in CSV format. The property indicates in which column index is every peptide data. The required columns are: Sequence, MS2, Charge, Id, RunId, SearchId, and TheoMz. Additionally for single quantification the noise, and the start and the end columns of the encoded peaks are required.
  In the configuration file, this property will be identified by the `input.csv.columns` and its value will be the column number of every required peptide data separated by space.
  The regular expression `(([0-9]+)( |\t|\f))6([0-9]+)` identifies the value.

  ```
  Example:
  #indicated as      Sequence MS2 Charge Id RunId SearchId TheoMz
   Noise sPeak ePeak
  input.csv.columns =    6      2    5    3    1      0       4
    7     8     9
  ```

Preference variables
In this section will be shown the preference variables that the user can configure in order to do some action or obtain some information.

Exists some preferences related to the area calculation. All these preferences will be identified in the configuration file by the area. prefix.

- Preferences.method: Method that will be used for the area calculation. The property key will be `area.method` and its value can be `TRAPEZOID` or `GAUSSIAN`.

- Preferences.isotope: isotope of the isotopic distribution that will be used in order to perform the quantification. The key will be `area.isotope` and its value a natural number starting by 0 (monoisotopic).

- Preference.peptideTreatment: There are different peptide treatments that can be performed. This properties will start by the `area.peptideTreatment` prefix and their value will be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase).

  - `BASIC`: indicates if no peptide treatment will be performed.
  - `SMOOTH`: indicates if the smooth peptide treatment will be performed.
  - `SMOOTHLIMiTS`: indicates if the smooth + search peptide limits treatment will be performed.
  - `PPMSCORE`: indicates if the PPM Score treatment will be performed.
  - `PPMSCORESMOOTH`: indicates if the PPM Score + Smooth peptide treatments will be performed.

  In a quantification execution, can be enabled more than one methodology. If it is done, more than one output will be written (for every enabled methodology a different output will be written).

  ```
  Example:
  area.method = TRAPEZOID
  area.isotope = 0
  area.peptideTreatment.BASIC = 1
  area.peptideTreatment.SMOOTH = 0
  area.peptideTreatment.SMOOTHLIMITS = f
  area.peptideTreatment.PPMSCORE = 0
  area.peptideTreatment.PPMSCORESMOOTH = 0
  ```

  The values will be recognized using the following regular expressions:

  ```
  yes:  ([yY]([eE][sS])?)|([tT]([rR][uU][eE])?)|1
  no:  ([nN]([oO])?)|([fF]([aA][lL][sS][eE])?)|0
  ```

There are more preferences:

- Preferences.timeFormat: Output time format (seconds, minutes or hours). In the configuration file will be represented by the `timeFormat` key, and its value could be: `SECONDS`, `MINUTES`, or `HOURS`.

```
Example:
timeFormat = MINUTES
```

- Preferences.variableMz: If enabled, the methodology used to find the peptide peaks is using a variable mz (the next peak will have a similar mz to the previous obtained peak, so that the window will be previousMz +/ ppm), otherwise will be static (theoretical mz).
  In the configuration file, the property will be identified by the `peakSearch.variableMz key`, and its value could be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase).

  ```
  Example:
  peakSearch.variableMz = false
  ```

- Preferences.similarIntensity: If enabled, the methodology used to choose the peak from a set of peaks inside the ppm window, will be get the one which have the most similar intensity to the previous obtained peak. Otherwise, will use the mz in order to choose the peak.
  In the configuration file, this property will be represented by the `peakSearch.similarIntensity` key, and its value could be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase).

  ```
  Example:
  peakSearch.similarIntensity = yes
  ```

- Preferences.originalMz: This property will be used only if `peakSearch.similarIntensity` is not enabled. If the property is enabled, the methodology used in order to choose the peak from a set of peaks within the ppm window, will be get the one which has the most similar mz to the theoretical mz. Otherwise, will be the peak with the most similar mz to the previous peak.
  In the configuration file, this property will be identified by the `peakSearch.similarTheoreticalMz` key, and its value could be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase).

  ```
  Example:
  peakSearch.similarTheoreticalMz = 1
  ```

- Preferences.useMSWindow: If enabled, will use a MS Scan window in order to find the peptide peaks, i.e. will search in the x previous and posterior scans. Otherwise, the search will not be limited. In the configuration file, the property will be represented by the `peptideSearch.useMSWindow`, and its value could be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase).

  Example:
  `peptideSearch.useMSWindow = n`

- Preferences.action: Action to execute. The property will be identified by the `program.action` key and its value will be `QUANT` for batch quantification, `XQUANT` for cross quantification, or `SQUANT` for single quantification.

  Example:
  `program.action = XQUANT`

- Preferences.outputColumns: Contains all the possible output columns. In the configuration file, will be represented by the `output.NAME` key, where `NAME` will be the column name. Its value will be y, yes, t, true, 1, or n, no, f, false, 0 (upper or lowercase), in order to indicate if must be shown.

  At this moment the possible outputs are: searchId, runId, peptideId, ms2Scan, noise, s/n, firstScan, lastScan, numScans, score, area, maxIntensity, maxIntensityScan, maxIntensityRetT, peaks, areaRatio, areaMinRatio, areaMaxRatio, intensityRatio, intensityMinRatio, and intensityMaxRatio.

  Example:
  ```
  output.searchId = true
  output.runId = true
  output.peptideId = true
  output.ms2Scan = true
  ```

  If an output does not appear in the configuration file, will take the default value (false).

Default values
Some variables will have a default value that will be used if the property is not found in the configuration file or its value is not correct.

The properties that do not have default values are: input and output variables, labels, static and dynamic modifications. Below are written the other properties default values.

```
Configuration.ppm = 5;
Configuration.isotopicDistPeaks = 2;
Configuration.msNoiseWindow = 20;
Configuration.mzNoiseWindow = 25;
Configuration.msPeptideWindow = 20;
Configuration.msPeptideScansNotFound = 2;
Configuration.minPeaks = 1;
Configuration.maxSmooth = 30;
Configuration.numberOfThreads = 1;
Configuration.xMsWindow = 75;

Preferences.trapezoid = true;
Preferences.method = TRAPEZOID;
Preferences.isotope = 0;
Preferences.peptideTreatment = BASIC;
Preferences.timeFormat = TimeFormat.MINUTES;
Preferences.variableMz = false;
Preferences.similarIntensity = true;
Preferences.originalMz = false;
Preferences.useMSWindow = false;
```

All output columns are set as false as default value.

13.5 Configuration file example

In this section a configuration file example for the proteomics quantification algorithm is shown:

```
;!!!! [ PREFERENCES ] !!!!!

; -------- Program action --------
program.action = QUANT

; -------- Area preferences --------
area.method = TRAPEZOID
area.isotope = 0
area.peptideTreatment.BASIC = 1
area.peptideTreatment.SMOOTH = 0
area.peptideTreatment.SMOOTHLIMITS = f
area.peptideTreatment.PPMSCORE = 0
area.peptideTreatment.PPMSCORESMOOTH = 0


timeFormat = MINUTES

; -------- Peptide search preferences --------
peakSearch.variableMz = false
peakSearch.similarIntensity = true
peakSearch.similarTheoreticalMz = false
peptideSearch.useMSWindow = false

; -------- Output columns --------
output.searchId = true
output.runId = true
output.peptideId = true
output.ms2Scan = true
output.noise = true
output.s/n = true
output.firstScan = true
output.lastScan = true
output.numScans = true
output.score = true
```

```
output.area = true
output.maxIntensity = true
output.maxIntensityScan = true
output.maxIntensityRetT = true
output.areaRatio = true
output.areaRatioMin = true
output.areaRatioMax = true
output.intensityRatio = true
output.intensityRatioMin = true
output.intensityRatioMax = true
output.findingScore = true;
output.peaks = false;


;!!!! [ PROGRAM CONFIGURATIONS ] !!!!!

[ LABELS ]
#label_Medium = K 4.02510698 #
label_Light = _ ; no marks
label_Heavy = K 8.01419881 # ;@

[ ADDITIONAL MASSES ]
mod_M* = 15.9949146221 1

[ CONSTANT MASSES ]
add_G_Glycine = 0.0000
add_A_Alanine = 0.0000
add_S_Serine = 0.0000
add_P_Proline = 0.0000
add_V_Valine = 0.0000
add_T_Threonine = 0.0000
add_C_Cysteine = 57.021464
add_L_Leucine = 0.0000
add_I_Isoleucine = 0.0000
add_X_LorI = 9000.0
add_N_Asparagine = 0.0000
add_B_avg_NandD = 0.0000
add_D_Aspartic_Acid = 0.0000
```

```
add_Q_Glutamine = 0.0000
add_K_Lysine = 0.0000
add_Z_avg_QandE = 0.0000
add_E_Glutamic_Acid = 0.0000
add_M_Methionine = 0.0000
add_O_Ornithine = 0.0000
add_H_Histidine = 0.0000
add_F_Phenyalanine = 0.0000
add_R_Arginine = 0.0000
add_Y_Tyrosine = 0.0000
add_W_Tryptophan = 0.0000

[ PEPTIDE configurations ]
mzPPM = 5                       ; +/- 5 ppm
isotopicDistPeaks = 2
msNoiseWindow = 20              ; +/- 20
mzNoiseWindow = 25              ; +/- 25
msPeptideWindow = 20            ; +/- 20
msPeptideScansNotFound = 2
minPeaks = 1
maxSmooth = 30

[ QUANTIFICATION configurations ]
numberOfThreads = 1

[ CROSS QUANTIFICATION configurations ]
xQuantMsWindow = 75

[ INPUT ]
; ------ NORMAL QUANTIFICATION ------
inputPeptidesFilePath = path/to/03218.csv
inputMsFilePath = path/to/03218.mzXML


; -------- Input columns --------
# indicated as      Sequence MS2 Charge Id RunId SearchId TheoMz
;Quant
input.csv.columns =    12      3   10    5    1      0       9
```

```
[ OUTPUT ]
outputFilePath = path/to/save/output/outputQuant.csv

outputImgPath = /path/to/save/images/ ; only folder
```